ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ
САНКТ-ПЕТЕРБУРГСКОЕ ОТДЕЛЕНИЕ
МАТЕМАТИЧЕСКОГО ИНСТИТУТА ИМ. В. А. СТЕКЛОВА РАН

На правах рукописи

Близнец Иван Анатольевич

Алгоритмы и нижние оценки на вычислительную сложность задач модификации графов

01.01.06 — математическая логика, алгебра и теория чисел

Диссертация
на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:
к.ф.-м.н. А. С. Куликов

Санкт-Петербург
2016

Оглавление

Введение

Данная работа посвящена изучению задач, связанных с модификацией графов. Задачи модификации графов играют важную роль в теоретической информатике и имеют множество приложений, включая молекулярную биологию [1, 2], вычисления с разреженными матрицами [3, 4], компьютерное зрение [5], реляционные базы данных [6] и другие. Данная диссертация содержит как доказательство условных нижних оценок, так и алгоритмы для решения задач модификации графов. Во введении описана мотивация, известные и полученные результаты исследования.

Общая формулировка рассматриваемых задач

В задачах модификации графов требуется в заданном графе произвести минимальное количество изменений с множеством вершин или рёбер, чтобы получить граф из заранее заданного класса Π. Задачи модификации графов бывают трёх типов: вершинные, рёберные и смешанные. В вершинных задачах разрешается производить изменения с множеством вершин. Наиболее часто рассматривают вершинные задачи со следующей формулировкой: требуется удалить минимальное количество вершин из заданного графа так, чтобы полученный граф принадлежал заданному классу графов Π. Классические примеры таких задач: задача о максимальной клике, вершинное покрытие, задача о разрезании контуров. В случае максимальной клики класс Π — это класс всех полных графов. В задаче вершинного покрытия класс Π — это класс всех пустых (безрёберных) графов. В задаче разрезания контуров класс Π состоит из ациклических графов. В рёберных задачах модификации графов производятся операции над множеством рёбер. В зависимости от рассматриваемой задачи мы можем только удалять рёбра, только добавлять ребра или же проделывать обе операции одновременно. Другими словами, для заданного графа $G = (V, E)$ нашей целью является найти множество рёбер $F \subset V \times V$ минимального размера так, чтобы граф $G' = (V, E \Delta F)$ принадлежал классу Π, где $E \Delta F \equiv (E \setminus F) \cup (F \setminus E)$. В задачах удаления мы дополнительно требуем, чтобы F было подмножеством E; в задачах дополнения — чтобы $F \cap E = \emptyset$; в задачах изменения нет ограничений на множество F, то есть мы можем как удалять, так и добавлять ребра. Примером подобной задачи является задача хордального дополнения, где требуется найти минимальный хордальный суперграф, то есть добавить минимальное число рёбер, чтобы граф стал хордальным. В смешанных задачах разрешено изменять множество вершин и рёбер одновременно.

Приложения

Задачи, связанные с модификацией графов, являются фундаментальными задачами теории графов. Уже в 1979 г. Гэри и Джонсон упомянули около двух десятков различных задач вершинной и рёберной модификации графов. Как уже было упомянуто, задачи модификации графов имеют приложения во многих областях: молекулярной биологии, вычислительной алгебре, реляционных базах данных и т.д. Часто с помощью графа представляют некоторые данные, полученные экспериментальным путём. В таких случаях добавление или удаление ребра является корректировкой допущенной ошибки, а удаление вершины может рассматриваться как устранение некоторых «выбросов» в данных, то есть тогда, когда мы предполагаем, что почти все данные, полученные в конкретной вершине, имеют неверное значение.

Одной из самых известных задач модификации графа является задача рёберного дополнения до хордального графа. Изучение хордального дополнения первоначально было вызвано его связью с методом Гаусса на разрежённых графах. Метод Гаусса применяется для решения линейных уравнений. При использовании данного метода производятся манипуляции над строками матрицы, представляющей систему уравнений. При проведении подобных операций некоторые нулевые элементы матрицы могут перестать быть такими, что потребует дополнительной памяти для хранения новых значений. Желательно применить метод Гаусса таким способом, при котором будет создано минимальное количество новых ненулевых элементов матрицы.

В [3] было показано, что если матрицу системы линейных уравнений рассмотреть как матрицу смежности графа $G = (V, E)$, то метод Гаусса эквивалентен процессу элиминации графа G. Если дан порядок $\alpha = (v_1, v_2, \ldots, v_n)$ вершин графа G, то процесс элиминации относительно порядка α происходит следующим образом: вершины удаляются одна за другой в порядке α, перед удалением вершины v смежные с ней вершины соединяются друг с другом (окрестность $N(v)$ превращается в клику). В [7] было доказано, что граф, полученный добавлением к G всех создаваемых таким образом рёбер, является хордальным, более того любой хордальный граф может быть получен таким способом. Получается, что задача минимизации количества новых ненулевых значений, созданных во время метода Гаусса, эквивалентна задаче дополнения соответствующего графа до хордального графа с использованием минимального числа рёбер.

Одним из наиболее распространённых приложений модификации до интервального графа является задача, возникающая в молекулярной биологии: задача о рёберном дополнении до интервального графа возникает при сборке генома человека. При анализе ДНК, как правило, имеется следующая информация: кусочки-интервалы (называемые клонами) из ДНК и экспериментальные данные о попарном наложении интервалов. Нашей целью является построение модели относительного расположения клонов. При проведении экспериментов возможно возникновение ошибок: некоторые пересечения могут быть не обнаружены. В этом случае граф, построенный по заданной информации, не будет интервальным. И для устранения ошибок следует добавить некоторое количество рёбер так, чтобы граф стал интервальным (каждое новое ребро будет сигнализировать о допущенной ошибке). Разумно предположить, что произошло небольшое количество ошибок и правильное расположение — это такое расположение, при котором требуется добавить минимальное число рёбер, чтобы граф стал интервальным.

Известные результаты

Задача о максимальном индуцированном П-подграфе (подграфе, принадлежащему классу графов П) является вершинной задачей модификации. В данной задаче для заданного фиксированного класса П в поданном на вход графе G требуется найти индуцированный подграф H из класса П с максимальным числом вершин. Льюис и Янакакис доказали следующие теоремы.

Теорема 1. [8] Если класс графов П обладает свойством наследственности и является нетривиальным (существует бесконечное количество графов, принадлежащих классу П, и бесконечное число классов, не принадлежащих классу П), то задача поиска максимального индуцированного П-подграфа является NP-трудной.

Теорема 2. [9] Для любого нетривиального свойства П, которое верно для любого связного индуцированного подграфа П-графа, задача поиска наибольшего связного индуцированного П-подграфа является NP-трудной.

В литературе встречается большое многообразие работ, рассматривающих задачу максимального индуцированного П-подграфа для некоторых конкретных классов П. С точки зрения точных экспоненциальных алгоритмов задача рассмотрена в работах [10, 11, 12, 13, 14, 15, 16, 17, 18], с точки зрения параметризованных алгоритмов — в [19, 20, 21, 22, 23, 24], приближенные алгоритмы и возможность приближения представлены в работах [19, 20, 25, 26]. В данной диссертации рассматриваются только подходы, позволяющие получить точное решение, то есть мы рассматриваем построение точных экспоненциальных и параметризованных алгоритмов.

Примеры изученных классов включают следующие: безрёберные, планарные, внешнепланарные, двудольные, полные двудольные, ациклические, с заданными ограничениями на степень и другие. С точки зрения точных алгоритмов, если принадлежность классу графов П может быть проверена за полиномиальное время, то задача тривиально решается полным перебором за время $2^n n^{O(1)} = O^*(2^n)$ ($O^*(\cdot)$ скрывает множители, полиномиально зависящие от размера входных данных) на графе, состоящем из n вершин. Однако для многих конкретных классов П задача может быть решена быстрее полного перебора. Так, известными примерами являются задачи, когда граф П — это класс безрёберных графов (задача о максимальном независимом множестве), класс ациклических графов (задача о максимальном индуцированном лесе), класс двудольных графов, планарных, d-вырождающихся, регулярных, графов кластеров или двудольных клик (см. таблицу 1).

Известно, что любой класс графов П, обладающий свойством наследственности, можно описать с помощью множества индуцированных запрещённых подграфов $\mathcal{F}$. Другими словами, $G \in \Pi$ тогда и только тогда, когда G не содержит в качестве индуцированного подграфа любой граф F из множества $\mathcal{F}$. Так, множество ациклических графов можно описать множеством $\mathcal{F}$, состоящим из всех циклов, в то время как множество хордальных графов — множеством всех циклов на четырёх и более вершинах. Множество графов-кластеров (граф состоящий их непересекающегося объединения полных графов) характеризуется множеством запрещённых индуцированных подграфов, состоящим из одного графа P_3. В том случае, когда для класса графов П множество запрещённых подграфов конечно, максимальный индуцированный П-подграф может быть найден значительно быстрее $O^*(2^n)$ с помощью простого алгоритма расщепления. В случае же, когда множество $\mathcal{F}$ бесконечно, задача становится нетривиальной и построение алгоритмов с временем работы c^n (где $c < 2$) становится весьма затруднительным, даже если множество $\mathcal{F}$ состоит из очень простых графов, например, циклов. Несмотря на нетривиальность построения таких алгоритмов, для многих естественных классов графов существуют алгоритмы с временем работы c^n, где $c < 2$. В связи с этим в работе [27] была сформулирована следующая гипотеза:

Гипотеза 1. Для любого полиномиально распознаваемого класса П со свойством наследственности задача максимального индуцированного подграфа может быть решена за время c^n, где $c < 2$ константа.

Таблица 1: Известные результаты для задачи о максимальном индуцированном П-подграфе

Класс графов	Время	Авторы
Безреберные	$\mathcal{O}(1.2109^n)$	Робсон [10]
Ацикличные (леса)	$\mathcal{O}(1.7548^n)$	Фомин и др. [11]
Двудольные	$\mathcal{O}(1.62^n)$	Раман и др. [12]
Планарные	$\mathcal{O}(1.7347^n)$	Фомин и др. [13]
d-вырожденные	$\mathcal{O}((2-\epsilon_d)^n)$	Пилипчук и Пилипчук [14]
Кластер графы	$\mathcal{O}(1.6181^n)$	Фомин и др. [15]
Биклики	$\mathcal{O}(1.3642^n)$	Гасперс и др. [16]
С $o(n/\log n)$ древесной шириной	$\mathcal{O}(1.7347^n)$	Виллангер и Фомин [17]
r-регулярные	$\mathcal{O}((2-\epsilon_r)^n)$	Гупта и др. [18]
Паросочетания	$\mathcal{O}(1.6957^n)$	Гупта и др. [18]

Опишем известные результаты для задач вершинной модификации с точки зрения параметризованной сложности. Параметризованная сложность рассматривает задачи, содержащие во входе специальное значение k, называемое параметром (как правило, k меньше размера входа n). Алгоритм для задачи называется FPT-алгоритмом, если время его работы не превышает $f(k)n^{O(1)}$, где f — произвольная вычислимая функция. Часто в качестве параметра выступает размер искомого решения или его верхняя/нижняя граница. Отметим, что для NP-трудных задач функция f не может быть полиномиальной в предположении неравенства классов **P** и **NP**, если значение параметра не превосходит размер входа. Главным вопросом для задач с параметром является наличие FPT-алгоритма. В случае существования такого алгоритма дальнейшие исследования посвящают построению алгоритмов со всё меньшими значениями функции f. Задачи вершинной модификации графов до класса Π естественным образом описываются с помощью задач с параметром: для заданного графа G и целого k требуется определить, можно ли удалить не более k вершин из графа так, чтобы полученный граф принадлежал классу Π.

Каи в работе [28] показал, что если класс Π описывается конечным числом запрещённых индуцированных подграфов, то для задачи вершинной модификации до класса Π существует FPT-алгоритм. Класс хордальных графов описывается бесконечным числом запрещённых графов и поэтому к нему не применима теорема Каи. Несмотря на это для задачи вершинной модификации до хордального графа существует FPT-алгоритм, как доказал Маркс в [24]. Другим подобным примером является задача вершинной модификации до интервального графа, для которой Као и Маркс построили алгоритм с временем работы $10^k n^{O(1)}$ [29]. В случае класса собственных интервальных графов Као привел алгоритм с временем работы $O(6^k(n+m))$ [30]. С другой стороны, в [31] было показано, что для случая, когда класс Π является классом совершенных графов или слабых хордальных графов, задача является $W[2]$-трудной, а значит, в предположении FPT$\neq W[1]$ не допускает FPT-алгоритма.

В задачах рёберной модификации разрешено удалять или добавлять ребра для получения графа из определённого класса. Приведём описание данного направления, представив результаты лишь для одной наиболее хорошо изученной задачи — дополнения до хордального графа. Впервые данная задача была описана в книге Гэри и Джонсона [32] в списке из 12 открытых задач, чей статус принадлежности классу NP-трудных задач неизвестен. Позднее Янакакис [33] доказал NP-трудность данной задачи. С точки зрения экспоненциальных алгоритмов задача изучена в работах [34, 35]. Для этой же задачи был построен приближенный алгоритм, выдающий решение, состоящее не более чем из $8 \cdot OPT^2$ ребер [36], где OPT — количество ребер в оптимальном решении. С точки зрения параметризованных алгоритмов задача о дополнении до хордального графа первоначально была изучена в работе [37], где был приведён алгоритм с временем работы $O(m16^k)$ (здесь и далее m — число ребер в графе), позднее данная оценка была улучшена в работе Каи [28] до $O((n+m)\frac{4^k}{k+1})$, и до $O(2.36^k+k^2mn)$ в работе Бодлайндера и др. Большим прорывом в области параметризованных алгоритмов было построение Виллангером и Фоминым субэкспоненциального алгоритма для этой задачи [38] с временем работы $(2^{O(\sqrt{k}\log k)} + k^2nm)$. Позднее появились субэкспоненциальные алгоритмы для задач рёберного дополнения до порогового [39], тривиально совершенного [39], псевдоразделённого [39], интервального [40], собственно интервального графов [41]. Стоит отметить, что в предположении гипотезы экспоненциального времени [42] (неформально говоря, гипотеза утверждает, что задача 3-Выполнимости не может быть решена за субэкспопенциальное от количества переменных время) субэкспоненциальные алгоритмы существуют не для всех задач о рёберных дополнениях. Если класс графов Π характеризуется множеством запрещённых графов $\mathcal{F}$, где $\mathcal{F} \in \{\{2K_2\}, \{C_4\}, \{P_4\}, \{2K_2, P_4\}\}$, то задача рёберного дополнения до класса Π не допускает алгоритма с временем работы $2^{o(k)}n^{O(1)}$ [39].

На данный момент в литературе случай общей (смешанной) задачи модификации (когда можно удалять вершины из графа, а также добавлять и удалять рёбра) изучен лишь в небольшом количестве статей. Так Маркс и Као показали [43], что если разрешено: удалить не более k_1 вершин, удалить не более k_2 рёбер, добавить не более k_3 рёбер, то заданный граф можно трансформировать в хордальный граф с помощью алгоритма с временем работы $2^{O(k\log(k))}n^{O(1)}$, где $k = k_1+k_2+k_3$. Совсем недавно аналогичный результат для класса собственных интервальных графов привел Као в работе [30].

Описание цели, полученных результатов и структуры диссертации

Цели работы.

1. Разработать алгоритм поиска максимального индуцированного хордального подграфа в заданном графе с временем работы меньше 2^n. Построить аналогичный алгоритм для поиска максимального индуцированного интервального подграфа.

2. Получить максимально точные нижние оценки на вычислительную сложность для задач о рёберном дополнении до класса графов Π. Рассмотреть случаи, когда класс графов Π является классом хордальных, интервальных, собственно интервальных, пороговых, тривиально совершенных графов.

Актуальность исследования подтверждается публикациями на данную тематику в престижных журналах и докладах на значимых конференциях, возрастающим интересом к данной области, проявляющемся в ежегодном увеличении публикаций, а также связью с многими практическими задачами, описанными ранее.

Методы исследований. В первой части работы, рассматривающей построение алгоритмов, использован метод расщепления, также используются различные характеризации класса хордальных графов. Во второй части работы, посвящённой нижним оценкам на вычислительную сложность, использованы методы доказательства нижних оценок с помощью гипотезы ETH. Также был применяется метод конвертации свойства неприближаемости одной задачи в нижнюю оценку на сложность вычисления для другой задачи.

Научная новизна. Все результаты диссертации являются новыми и ранее неизвестными.

Теоретическая и практическая ценность. Работа носит теоретический характер. Её результаты могут быть использованы как для построения новых алгоритмов, так и для доказательства нижних оценок на вычислительную сложность.

Положения, выносимые на защиту.

- Алгоритм для поиска наибольшего индуцированного хордального/интервального подграфа.
- Нижняя оценка на вычислительную сложность для задач рёберного дополнения в предположении гипотезы экспоненциального времени.
- Нижняя оценка на вычислительную сложность для задач рёберного дополнения в предположении гипотезы о не существовании субэкспоненциальной приближающей схемы для задачи о минимальной бисекции на графах с ограниченной степенью.

Достоверность результатов и апробация работы. Достоверность результатов обеспечивается их строгим математическим доказательством. Результаты диссертационной работы были изложены на следующих конференциях и семинарах:

1. Международная конференция “21st European Symposium on Algorithms ESA 2013” (София Антиполис, Франция, ESA 2013).
2. Санкт-Петербургский городской семинар по дискретной математике (Санкт-Петербург, Россия, 2013).
3. Семинар алгоритмической группы университета Бергена (Берген, Норвегия, 2013).
4. Конференция “Satisfiability Lower Bounds and Tight Results for Parameterized and Exponential-Time Algorithms”, Simons University (Беркли, США, 2015).
5. Конференция “Problems in Theoretical Computer Science” (Москва, Россия, 2015).

6. Международная конференция “Annual ACM-SIAM Symposium on Discrete Algorithms” (Арлингтон, США, SODA 2016).

Публикации. Основные результаты диссертации опубликованы в рецензируемых научных изданиях — [44], [27], [45].

Работы [44], [27], [45] написаны в соавторстве. В работе [44] диссертанту принадлежит алгоритм построения решения в случае наличия в графе большой клики (Лемма 1), теоремы 1-2 получены в неразрывном сотрудничестве. В работе [45] диссертанту принадлежат: доказательство нижней оценки на сложность для задачи об оптимальном линейном упорядочивании в предположении справедливости гипотезы ETH (теоремы 3.8 и 1.5), а также доказательство нижних оценок для задач о хордальном, интервальном, собственно интервальном, цепочном, тривиально совершенном и пороговом дополнениях, основыванных на нижней оценке на сложность для задачи об оптимальном линейном упорядочивании (теоремы 1.1 и 1.3). Нижняя оценка на вычислительную сложность для задачи об оптимальном линейном упорядочивании, в предположении гипотезы о сложности приближения задачи о минимальной бисекции, первоначально была получена диссертантом и впоследствии упрощена соавторами (теорема 1.6, леммы 4.1-4.4).

Полученные результаты представляют собой алгоритм для задачи вершинной модификации и условные нижние оценки на вычислительную сложность для задачи рёберного дополнения до определённых классов графов.

В главе 1 представлены требуемые предварительные сведения, определения и обозначения, использующиеся в последующих главах.

В главе 2 приводится алгоритм для задачи о максимальном индуцированном Π-подграфе, где класс Π — заранее зафиксированный класс графов, удовлетворяющий четырём свойствам:

Свойство 1. Класс Π — подкласс хордальных графов со свойством наследственности.

Свойство 2. Все графы из множества $\mathcal{F}_\Pi$ связны и не содержат клики размера $\aleph + 1$ (для некоторой константы $\aleph$).

Свойство 3. Принадлежность графа классу Π можно определить за полиномиальное время. Другими словами, класс Π полиномиально распознаваем.

Свойство 4. Существует алгоритм $\mathcal{A}$, принимающий на вход граф G вместе с кликой $S \subset V(G)$. Алгоритм выдаёт «ДА» или «НЕТ» так, что выполнены следующие условия:

- Пусть $\mathcal{A}$ выдаёт «ДА» на входах (G_1, S_1) и (G_2, S_2), где $|S_1| = |S_2|$. Тогда граф G', получающийся при склеивании графов G_1 и G_2, так что каждая вершина S_1 отождествляется ровно с одной вершиной из S_2, должен принадлежать классу Π.

- Если $G \in \Pi$, тогда существует разделитель-клика S такая, что $V(G) \setminus S$ можно разбить на два множества X_1 и X_2, удовлетворяющих следующим свойствам: (i) $|X_1|, |X_2| \leq \frac{2}{3}V(G)$, (ii) $E(X_1, X_2) = \emptyset$, (iii) алгоритм $\mathcal{A}$ выдаёт «ДА» на входах $(G[X_1 \cup S], S)$ и $(G[X_2 \cup S], S)$.

Построенный алгоритм является первым алгоритмом, работающим быстрее полного перебора для таких классов графов как хордальные и интервальные. Главным результатом главы 2 является следующая теорема:

Теорема 3. Пусть $\mathcal{F}$ — конечное множество графов и Π — класс графов, удовлетворяющий свойствам (1)–(4). Существует алгоритм, который в заданном графе G на n вершинах находит $\mathcal{F}$-свободный индуцированный Π-подграф за время $\mathcal{O}^*(2^{\lambda n})$ для некоторого $\lambda < 1$, где λ зависит только от $\aleph$ и $\mathcal{F}$.

Данную теорему можно рассматривать как частичный прогресс на пути к доказательству гипотезы 1.

В главе 3 приведены условные нижние оценки на сложность вычисления рёберного дополнения до хордального, интервального, собственно интервального, порогового и совершенно тривиального графов. Основные результаты главы получены в предположении гипотезы ETH и следующей гипотезы.

Гипотеза 2. Существуют вещественные числа $0 \leq \alpha < \beta \leq 1$ и натуральное число $d \geq \frac{4}{\beta-\alpha}$ такие, что для задачи о минимальной бисекции$(d)_{[\alpha,\beta]}$ не существует алгоритма с временем работы $2^{o(n)}$.

Основным результатом главы является следующая теорема:

Теорема 4. Если гипотеза 2 верна, тогда не существует алгоритма для задачи цепочного дополнения с временем работы $2^{o(n+m)}$, а для задач хордального, интервального, собственно интервального, порогового, тривиально совершенного дополнения не существует алгоритмов с временем работы $2^{o(n)}$. Таким образом ни одна их этих задача не может быть решена за время $2^{o(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$.

Также в главе 3 представлены доказательства следующих теорем:

Теорема 5. Если гипотеза 2 верна, тогда существует такое $d \in \mathbb{N}$, что для задачи ОЛУ$_{\leq}(d)$ на мультиграфах не существует алгоритма с временем работы $2^{o(n)}$.

Теорема 6. Если гипотеза ETH верна, тогда существует $c \geq 1$ такое, что задачи хордального, интервального, собственно интервального, порогового, тривиально совершенного и цепочного дополнения не допускают алгоритмов с временем работы $2^{\mathcal{O}(\sqrt{n}/\log^c n)}$, а значит и с временем $2^{\mathcal{O}(k^{1/4}/\log^c k)} \cdot n^{\mathcal{O}(1)}$.

Глава 1

Предварительные сведения

1.1 Графы

В данной работе мы используем стандартные обозначения из теории графов. *Граф* G — это пара $(V(G), E(G))$, где $V(G)$ — это множество *вершин*, а $E(G) \subseteq \binom{V(G)}{2}$ — множество *рёбер* (в диссертации рассматриваются только неориентированные графы). Будем использовать обозначения V и E для множества вершин и рёбер графа G, если из контекста ясно, какой граф рассматривается. Граф H является *подграфом графа* G, если $V(H) \subseteq V(G)$ и $E(H) \subseteq E(G)$. Граф H является *индуцированным подграфом* G, если $V(H) \subseteq V(G)$ и $E(H) = E(G) \cap \binom{V(H)}{2}$. Индуцированный подграф графа G с множеством вершин X обозначим через $G[X]$. Через $G \setminus X$ обозначим подграф G, индуцированный множеством вершин $V(G) \setminus X$. Будем говорить, что подмножество вершин $W \subseteq V$ *связно*, если граф $G[W]$ связен.

Дополнение графа G — это граф с множеством вершин V и множеством рёбер $\binom{V}{2} \setminus E$, обозначим дополнение через $\overline{G}$. Для $X \subseteq V$ обозначим через $\delta_G(X)$ множество рёбер, исходящих из X (содержащих ровно одну вершину в X), индекс G опускается, если из контекста ясно какой граф используется. Если $X, Y \subseteq V$ не пересекающиеся множества, то $E_G(X, Y)$ обозначает множество рёбер, идущих из X в Y. Через

$G[X,Y]$ обозначим *индуцированный двудольный подграф* G *с долями* X *и* Y. То есть множество вершин графа $G[X,Y]$ равняется $X \cup Y$, а множество рёбер — $E_G(X,Y)$. Определим *разрез* как множество рёбер $E_G(A,B)$ для некоторого разбиения (A,B) множества V. Размер разреза считаем равным $|E_G(A,B)|$.

Размер множеств V и E обозначим через n и m соответственно. $\deg_G(v)$ — степень вершины v (число инцидентных рёбер). Граф называется *d-регулярным*, если степень каждой вершины равняется d. Максимальную степень графа G обозначим через Δ_G. Множество $N_G(v) = \{w : \{v,w\} \in V(G)\}$ — это множество соседей (открытая окрестность вершины) вершины v. Расширим данное обозначение на подмножества вершин X, положим $N_G(X) = \bigcup_{v \in X} N_G(v) \setminus X$. Замкнутую окрестность множества X обозначим через $N_G[X] = N_G(X) \cup X$. Значение нижних индексов будет опущено (будем писать просто $\deg(v), \Delta, N(X), \delta(X), E(U,V)$) тогда, когда из контекста ясно, какой граф рассматривается.

Класс графов Π — это семейство графов. Говоря Π-граф или Π-подграф, мы имеем в виду, что рассматриваемый граф или подграф принадлежит классу Π. Мы говорим, что класс графов обладает свойством *наследственности*, если класс Π замкнут относительно взятия индуцированных подграфов. Любой класс графов со свойством наследственности может быть описан списком (возможно, бесконечным) минимальных запрещённых подграфов $\mathcal{F}_\Pi$: граф $G \in \Pi$ тогда и только тогда, когда он не содержит никакого индуцированного подграфа из множества $\mathcal{F}_\Pi$, и для каждого графа $H \in \mathcal{F}_\Pi$ любой его индуцированный подграф, кроме самого H, принадлежит классу Π. Граф, не содержащий никакой индуцированный подграф из списка $\mathcal{F}$, будем называть *$\mathcal{F}$-свободным графом*. В таблице 1.1 приведены характеризации-определения рассматриваемых классов графов с помощью запрещённых индуцированных подграфов (см. рис. 1.1).

Таблица 1.1: Характеризации-определения различных классов графов.

Класс графов	Запрещённые подграфы
Хордальные	$C_n, n > 3$
Интервальные	$C_n, n > 3$; n-палатка, $n \geq 3$; n-сеть, $n \geq 2$; зонт; двудольная клешня
Собственно интервальные	$C_n, n > 3$; 3-палатка; 2-сеть; клешня
Пороговые	$C_4, P_4, 2K_2$
Тривиально совершенные	C_4, P_4
Разделяемые	$C_4, C_5, 2K_2$
Птолемея	$C_n, n > 3$; самоцвет
Блоковые	$C_n, n > 3$; алмаз

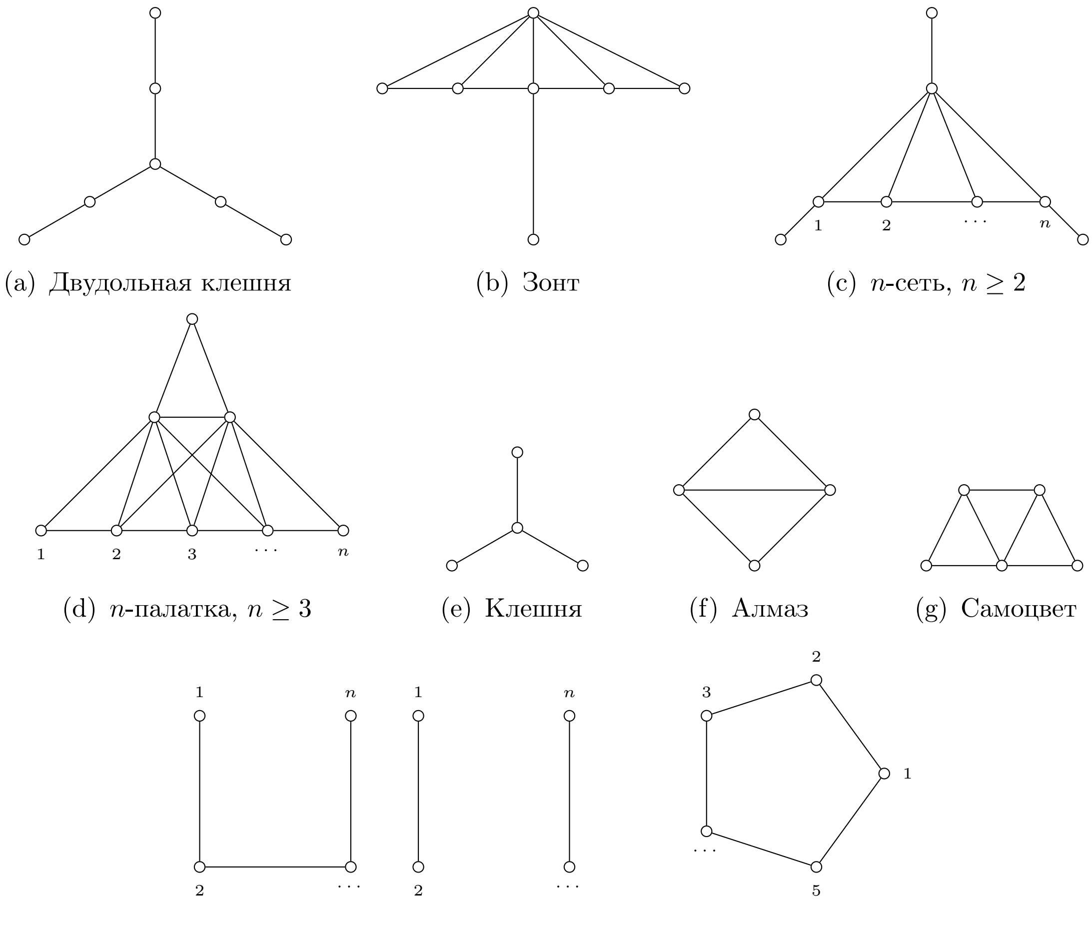
1.2 Вычислительная сложность

Параметризованная задача Q — это подмножество $\Sigma^* \times \mathbb{N}$ для некоторого фиксированного конечного алфавита Σ. Экземпляр задачи Q — это пара $(x, k) \in \Sigma^* \times \mathbb{N}$, где целое k называется параметром. Алгоритм для параметризованной задачи называется *FPT-алгоритмом* (fixed parameter tractable), если на любом экземпляре задачи (x, k) время его работы не превосходит $f(k) \cdot \text{poly}(|x|)$, где f — некоторая вычислимая функция. Такое время работы принято обозначать как $O^*(f(k))$. Здесь $O^*(\cdot)$ скрывает множители, зависящие от размера входных данных полиномиально.

Обозначим полиномиальное по времени детеминированное сведение от задачи распознавания X к задаче распознавания Y через $X \leq_{\mathrm{P}}^{\mathrm{lin}} Y$, если заданный экземпляр I задачи X преобразуется алгоритмом в экземпляр I' задачи Y, $|I'| = O(|I|)$ и ответы для экземпляров I, I' совпадают.

1.3 Выполнимость

Для задач выполнимости булевых формул в конъюнктивной нормальной форме (КНФ), далее просто задача выполнимости, в работе используются стандартные обозначения. Формула в КНФ представляет собой конъюнкцию клозов, клоз есть дизъюнкция литералов, а литерал есть булева переменная или её отрицание. Через $x_1, \ldots, x_n$ обозначаются перемен-



(a) Двудольная клешня (b) Зонт (c) n-сеть, $n \geq 2$

(d) n-палатка, $n \geq 3$ (e) Клешня (f) Алмаз (g) Самоцвет

(h) P_4 (i) $2K_2$ (j) C_n цикл (дыра)

Рис. 1.1: Запрещенные индуцированные подграфы для различных классов графов.

ные, а через $C_1, \ldots, C_m$ — клозы рассматриваемой формулы. Формула КНФ называется также l-КНФ формулой, если каждый клоз содержит не более l литералов, и формула КНФ является Pl-КНФ, если каждый клоз содержит ровно l литералов. *Означиванием* называется набор значений рассматриваемых булевых переменных. Означивание переменных $x_1, ..., x_n$ выполняет l-КНФ формулу, если каждый клоз содержит литерал со значением 1 и симметрично выполняет, если каждый клоз содержит литерал со значением 1 и литерал со значением 0. Заметим, что если означивание симметрично выполняет какой-то клоз, то его отрицание также симметрично выполняет этот клоз.

1.4 Гипотеза экспоненциального времени (ETH)

Гипотеза экспоненциального времени (ETH, exponential time hypothesis), предложенная Импальяцио, Патури и Зэйном [46, 47] широко применяется при доказательстве условных нижних оценок на вычислительную сложность параметризованных задач. В работе [48] представлен обзор нижних оценок, основанных на ETH-гипотезе. Неформально говоря ETH-гипотеза утверждает, что задача 3-Выполнимости не может быть решена за субэкспопенциальное от количества переменных время.

Гипотеза 1.1 (Гипотеза экспоненциального времени (ETH) [46, 47])**.** Пусть $s_3 = \inf_\delta\{\delta|$ существует алгоритм с временем $O(2^{\delta n})$ для задачи 3-ВЫП$\}$. Тогда $s_3 > 0$.

При доказательстве нижних оценок на вычислительную сложность для NP-трудных задач вместе с ETH гипотезой часто используется лемма о спарсификации:

Лемма 1.1 (Лемма о спарсификации, [46]))**.** Для любого $\epsilon > 0$ существует константа c_ϵ и алгоритм, который на вход принимает формулу ϕ в 3-КНФ с n переменными и возвращает l формул $\phi_1, ..., \phi_l$ 3-КНФ так, что выполнены следующие условия:

- $l = \mathcal{O}(2^{\epsilon n})$.
- Для любого i, ϕ_i содержит n переменных и каждая переменная содержится не более чем в c_ϵ дизъюнктах формулы ϕ_i.
- Формула ϕ выполнима тогда и только тогда, когда хотя бы одна формула ϕ_i выполнима.
- Время работы алгоритма не превосходит $\mathcal{O}^*(2^{\epsilon n})$.

Из гипотезы ETH и леммы о спарсификации вытекает субэкспоненциальная нижняя оценка на вычислительную сложность алгоритмов для задачи 3-Выполнимости относительно числа дизъюнктов:

Теорема 1.1 ([46])**.** Если гипотеза ETH верна, то не существует алгоритма для задачи 3-Выполнимости с временем работы $2^{o(m+n)}$, где n — это число переменных, а m — число дизъюнктов в формулах поданных на вход.

1.5 Задачи с зазором

В задачах с зазором известно, что поданный на вход экземпляр принадлежит одному из описанных языков. Требуется определить, какому именно из языков принадлежит анализируемый экземпляр. В задачах с зазором присутствует два параметра α, β. В частности, в задаче о максимальном разрезе с зазором$_{[\alpha,\beta]}$ необходимо отличить случай, когда граф содержит разрез размера не менее βm, и случай, когда заданный граф не содержит разреза содержащего более αm рёбер. Аналогично для задачи выполнимости с зазором $[\alpha, \beta]$ необходимо отличить формулы, в которых можно выполнить не менее βm дизъюнктов, и формулы, в которых нельзя выполнить более αm дизъюнктов.

1.6 Формулировки задач

Оптимальное линейное упорядочивание (ОЛУ)
Вход: Граф $G = (V, E)$, целое k.
Вопрос: Существует ли линейное упорядочивание π графа G стоимостью не больше k (т.е. $\Sigma_{\{u,v\}\in E(G)}|\pi(u) - \pi(v)| \leq k$)?

Оптимальное линейное упорядочивание$_{\leq}(d)$ (ОЛУ$_{\leq}(d)$)
Вход: Граф $G = (V, E)$ с вершинами степени не больше d, целое число k.
Вопрос: Существует ли линейное упорядочивание π графа G со стоимостью не более k (т.е. $\Sigma_{\{u,v\} \in E(G)} |\pi(u) - \pi(v)| \leq k$)?

Максимальный разрез
Вход: Граф $G = (V, E)$, целое число k.
Вопрос: Существует ли разрез, содержащий не менее k рёбер?

Максимальный разрез с зазором$_{[\alpha,\beta]}$
Вход: Граф $G = (V, E)$.
Случай 1: G содержит разрез размера не менее βm рёбер.
Случай 2: Размер любого разреза графа G не превосходит αm.

Минимальная бисекция
Вход: Граф $G = (V, E)$ с чётным количеством вершин, целое число k.
Вопрос: Существует ли разрез (A, B) размера не более k, такой что $|A| = |B|$?

Минимальная бисекция с зазором$(d)_{[\alpha,\beta]}$
Вход: d-регулярный граф $G = (V, E)$ с чётным числом вершин.
Случай 1: G не допускает разрезов (A, B) с числом рёбер меньше βm и $|A| = |B|$.
Случай 2: G содержит разрез (A, B) размера меньше αm, где $|A| = |B|$.

1.6.1 Задачи выполнимости

Р3-ВЫП

Вход: Р3-КНФ формула $\phi = C_1 \wedge \ldots \wedge C_m$, где $C_1, C_2, \ldots, C_m$ — дизъюнкты длины ровно три.

Вопрос: Существует ли означивание переменных формулы ϕ, выполняющее формулу (значение формулы на заданном наборе равняется 1)?

Р4-НВ-ВЫП

Вход: Р4-КНФ формула $\phi = C_1 \wedge \ldots \wedge C_m$, где $C_1, C_2, \ldots, C_m$ — дизъюнкты длины ровно четыре.

Вопрос: Существует ли выполняющий набор для формулы, такой что в любом дизъюнкте не все литералы принимают одно и то же значение?

Заз-Р3-ВЫП$_{[\alpha,\beta]}$

Вход: Р3-КНФ формула $\phi = C_1 \wedge \ldots \wedge C_m$, где $C_1, C_2, \ldots, C_m$ — дизъюнкты длины ровно три.

Случай 1: В ϕ можно выполнить одновременно не менее βm клозов.

Случай 2: В ϕ невозможно выполнить более αm клозов.

Заз-Р4-НВ-ВЫП$_{[\alpha,\beta]}$

Вход: Р4-КНФ формула $\phi = C_1 \wedge \ldots \wedge C_m$, где $C_1, C_2, \ldots, C_m$ — дизъюнкты длины ровно четыре.

Случай 1: Существует такое означивание, что в ϕ есть не менее βm выполненных дизъюнктов и каждый из них содержит как минимум один невыполненный литерал.

Случай 2: В ϕ невозможно выполнить более αm дизъюнктов так, чтобы каждый из этих αm дизъюнктов содержал невыполненный литерал.

1.6.2 Задачи реберного дополнения

Следующая задача задаёт общий вид задач дополнения до определённого класса графов X.

X-Дополнение
Вход: Граф G, целое число k.
Вопрос: Можно ли к графу G добавить k новых рёбер так, чтобы полученный граф принадлежал классу графов X?

Будем рассматривать следующие задачи рёберного дополнения: хордальное дополнение, цепочное дополнение, собственно интервальное дополнение, интервальное дополнение, пороговое дополнение, тривиально совершенное дополнение.

Глава 2

Наибольший индуцированный хордальный и интервальный подграфы

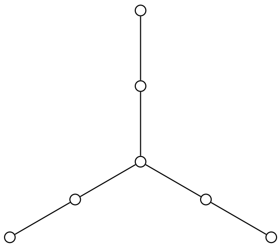
В данной главе приведён алгоритм поиска наибольшего хордального и интервального подграфов быстрее полного перебора. Более того, алгоритм также применим для поиска наибольших подграфов из других классов графов, удовлетворяющих определённым свойствам.

2.1 Используемые леммы

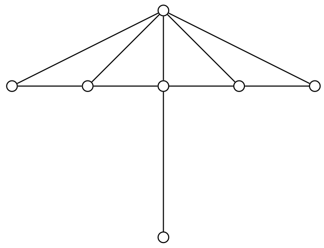
Предложение 2.1 ([49])**.** Если H — хордальный граф, то существует клика S в H и разбиение множества $V(H) \setminus S$ на два множества X_1, X_2, такие что:

- $|X_1|, |X_2| \leq \frac{2}{3}|V(H)|$,
- $E(X_1, X_2) = \emptyset$.

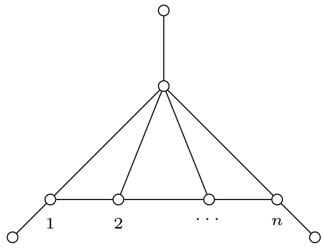
Такое множество S называется $\frac{2}{3}$-*сбалансированной разделяющей кликой* в H. Предложение 2.1 следует из аналогичного утверждения для деревьев и свойств древесного разложения хордальных графов (полное доказательство данного факта можно найти в книге [50]). Заметим, что поскольку $|X_2| \leq \frac{2}{3}|V(H)|$, то $|X_1| = |V(H)| - |S| - |X_2| \geq \frac{1}{3}|V(H)| - |S|$. Аналогичное неравенство верно для X_2.



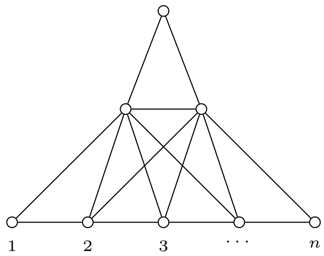
(a) Двудольная клешня



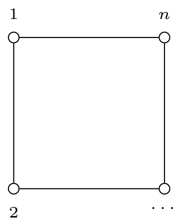
(b) Зонт



(c) n-сеть, $n \geq 2$



(d) n-палатка, $n \geq 3$



(e) C_n (цикл на n вершинах), $n \geq 4$

Рис. 2.1: Запрещённые индуцированные подграфы для класса интервальных графов.

Опишем классические результаты, которые нам потребуются при построении алгоритма. Следующий результат вытекает из наблюдения, что при расщеплении по запрещённому подграфу константного размера получается алгоритм с временем работы быстрее 2^n.

Лемма 2.1. Пусть множество $\mathcal{F}$ содержит конечное число графов и каждый граф из $\mathcal{F}$ содержит не более ℓ вершин. Пусть Π — это полиномиально распознаваемый класс графов, обладающий свойством наследственности. Предположим, что существует алгоритм $\mathcal{A}$, который для данного $\mathcal{F}$-свободного графа G на n вершинах за время $\mathcal{O}^*(2^{\epsilon n})$ находит максимальный индуцированный Π-подграф в графе G при некотором $\epsilon < 1$. Тогда существует алгоритм $\mathcal{A}'$, который для данного графа G на n вершинах находит максимальный индуцированный $\mathcal{F}$-свободный

П-подграф в графе G за время $\mathcal{O}^*(2^{\epsilon' n})$, где $\epsilon' < 1$ — это некоторая константа, зависящая от ϵ и ℓ.

Доказательство. Пусть Π' — это класс $\mathcal{F}$-свободных П-графов; поскольку ℓ — это константа, то класс Π' распознаваем за полиномиальное время. Алгоритм $\mathcal{A}'$ для заданного графа $G = (V, E)$ с n вершинами пытается найти максимальный индуцированный Π'-подграф, используя стандартный метод расщепления. В любой момент алгоритм хранит два непересекающихся множества $A, D \subseteq V$; в начальный момент $A = D = \emptyset$. При заданных A, D алгоритм пытается найти такое множество X максимального размера, что вершины из X индуцируют Π'-подграф, удовлетворяющий условиям $A \subseteq X$ и $D \cap X = \emptyset$. Как только достигается ситуация, что $|A \cup D| > (1 - \epsilon)n$, процедура расщепления останавливается и начинается перебор всевозможных вариантов расположения (принадлежат или не принадлежат максимальному индуцированному подграфу) для вершин из множества $V \setminus (A \cup D)$. То есть мы перебираем все подмножества $A' \subseteq V \setminus (A \cup D)$ и проверяем, является ли граф $G[A \cup A']$ Π'-подграфом. Для этого нам потребуется $\mathcal{O}^*(2^{|V \setminus (A \cup D)|}) \leq \mathcal{O}^*(2^{\epsilon n})$ времени.

На каждом шаге расщепляющей процедуры за полиномиальное время происходит проверка, содержит ли граф $G \setminus D$ индуцированный подграф из множества $\mathcal{F}$. Предположим, такой подграф найден, и пусть $S \subseteq V \setminus D$ — это множество его вершин. Очевидно, любой индуцированный Π'-подграф не содержит, как минимум, одну из вершин множества S. Поскольку вершины $S \cap A$ обязаны принадлежать решению в этой ветви расщепления, далее расщепляемся по вершинам из множества $S \setminus A$. Более точно для любого разбиения (A', D') множества $S \setminus A$, где $D' \neq \emptyset$, создается ветвь расщепления, в которой множество A' добавляется к A, а D' добавляется к D. Заметим, что тем самым мы создаём не более $2^{\ell'} - 1$ ветвей расщепления и увеличиваем $|A \cup D|$ на ℓ', где $\ell' = |S \setminus A| \leq \ell$. Более того, поскольку $\ell' \leq \ell$, то $2^{\ell'} - 1 \leq 2^{\epsilon_\ell \ell'}$ для некоторого $\epsilon_\ell < 1$, зависящего от ℓ.

Предположим, что $G \setminus D$ не содержит индуцированных подграфов из множества $\mathcal{F}$, то есть граф $G \setminus D$ является $\mathcal{F}$-свободным. В таком случае применяется алгоритм $\mathcal{A}$ к $G \setminus D$ для вычисления максимального индуцированного П-подграфа в графе $G \setminus D$. Поскольку граф $G \setminus D$ является $\mathcal{F}$-свободным, то найденный в нём подграф принадлежит классу Π'. Отметим, что в этот момент требование того, что искомый подграф содержит множество вершин A, ослабляется. Заметим, что подобное ослабление не влияет на корректность алгоритма, ведь найденный подграф является индуцированным Π'-подграфом графа G, а значит, может быть только больше первоначально искомого подграфа в данной ветви расщепления. Время работы, требуемое алгоритму $\mathcal{A}$ в данном случае, составляет $\mathcal{O}^*(2^{\epsilon|V \setminus D|}) \leq \mathcal{O}^*(2^{\epsilon n})$.

Оценим время работы алгоритма. Заметим, что в момент применения перебора выполнено $(1-\epsilon)n + \ell \geq |A \cup D| > (1-\epsilon)n$, поскольку с каждым шагом $|A \cup D|$ возрастает не более, чем на ℓ. При расщеплении значение $|A \cup D|$ увеличивается на ℓ' и создаётся не более $2^{\epsilon_\ell \ell'}$ ветвей, таким образом общее число ветвей, в которых применён алгоритм $\mathcal{A}$ или полный перебор, не более $2^{\epsilon_\ell((1-\epsilon)n+\ell)} = \mathcal{O}(2^{\epsilon_\ell(1-\epsilon)n})$. Запуск перебора или алгоритма $\mathcal{A}$ в каждом случае занимает не более $\mathcal{O}^*(2^{\epsilon n})$ времени. А значит, общее время работы алгоритма составляет $\mathcal{O}^*(2^{\epsilon' n})$, где $\epsilon' = \epsilon_\ell(1-\epsilon) + \epsilon < (1-\epsilon) + \epsilon = 1$.

□

Будем пользоваться следующей леммой Фомина и Виллангера [51] для перебора связных множеств вершин, при этом несущественно увеличивая время работы всего алгоритма.

Лемма 2.2 ([51])**.** Рассмотрим граф $G = (V, E)$. Для любой вершины $v \in V$ и любых целых чисел $b, f \geq 0$ число связных подмножеств $B \subseteq V$ таких, что

- $v \in B$,
- $|B| = b + 1$,
- $|N(B)| = f$,

не превосходит $\binom{b+f}{b}$. Более того, все такие подмножества можно перенумеровать за время $\mathcal{O}^*(\binom{b+f}{b})$.

Одновременно с леммой 2.2 будем использовать следующую оценку на значение биномиальных коэффициентов через значение двоичной энтропии.

Предложение 2.2. $\binom{n}{k} \leq 2^{H(\frac{k}{n})n}$ для любого $n \geq 1$ и $1 \leq k \leq n-1$, где $H(t) = -t\log_2 t - (1-t)\log_2(1-t)$.

Отметим, что из предложения 2.2 следует, что для константы $0 \leq \alpha \leq 1, \alpha \neq 1/2$ справедливо равенство $\binom{n}{\alpha n} = \mathcal{O}(2^{\kappa n})$ для некоторого $\kappa < 1$.

Последняя лемма, которая нам потребуется, основана на классической идее Шроупла и Шамира [52]. Первоначально идея была применена для задачи о сумме подмножеств при сведении её к задаче о 2-табличной сумме. В задаче о 2-Табличной сумме нам дано две матрицы T_i, $i = 1, 2$ размера $k \times m_i$ и вектор $\vec{s} \in \mathbb{Q}^k$. Столбцы каждой матрицы представляют собой m_i векторов из пространства $\mathbb{Q}^k$. В задаче требуется определить, существуют ли столбец из первой матрицы и столбец из второй матрицы такие, что их сумма равняется $\vec{s}$. Тривиальный алгоритм для задачи о 2-табличной сумме перебирает всевозможные пары векторов. Однако эта задача может быть решена значительно быстрее. Отсортируем столбцы T_1 в лексикографическом порядке за время $\mathcal{O}(km_1 \log m_1)$. После этого для каждого столбца $\vec{v}$ из T_2 проверим, содержит ли T_1 столбец $\vec{s} - \vec{v}$. Используя бинарный поиск, можно затратить не более $\mathcal{O}(k \log m_1)$ времени.

Лемма 2.3 ([52])**.** Задача о 2-табличной сумме может быть решена за время $\mathcal{O}((m_1 + m_2)k \log m_1)$.

2.2 Требуемые свойства

В данной главе приводится алгоритм для поиска в заданном графе индуцированного П-подграфа, где класс графов П удовлетворяет некоторым свойствам. В данном разделе описаны все требуемые для корректной работы алгоритма свойства класса Π. Будем рассматривать исключительно подклассы хордальных графов, обладающие свойством наследственности. Поэтому самое первое свойство, которое требуется от класса Π, таково.

Свойство (1.) Класс Π — подкласс хордальных графов со свойством наследственности.

Поскольку класс Π обладает свойством наследственности, то Π может быть описан множеством минимальных по включению запрещённых подграфов. Обозначим такое множество запрещённых подграфов через $\mathcal{F}_\Pi$. Потребуется следующее ограничение на множество $\mathcal{F}_\Pi$:

Свойство (2.) Все графы в множестве $\mathcal{F}_\Pi$ связны и не содержат клики размера $\aleph + 1$ для некоторой константы $\aleph$.

Напомним, что для класса Π хордальных графов $\mathcal{F}_\Pi$ состоит из циклов длины не меньше 4, поэтому $\aleph$ можно положить равным 2. Если же Π — это класс интервальных графов, то анализ рисунка 2.1 (с изображением всех запрещённых подграфов) показывает, что достаточно взять $\aleph = 4$. Далее будем обращаться с $\aleph$, как с некоторой универсальной константой для класса Π, выбор значений других констант в будущем может зависеть от значения $\aleph$. Время работы также будет зависеть от значения $\aleph$. Это касается как полиномиальных множителей, так и значений экспоненты в выражении, описывающем требуемое время работы.

Примером подкласса хордальных графов, не удовлетворяющего свойству (2), может служить класс строго хордальных графов, поскольку минимальный запрещённый подграф для этого класса может содержать сколь угодно большую клику. Другими примерами классов, не удовлетворяющих свойству (2), могут служить классы, у которых множество минимальных запрещённых подграфов содержит несвязные графы. Так, класс разделяемых графов (split graphs) не удовлетворяет свойству (2). Однако если количество несвязных графов в $\mathcal{F}_\Pi$ конечно (как в случае разделяемых графов), то можно применить лемму 2.1 для построения алгоритма.

Следующее свойство, которое нам понадобится, касается полиномиальной распознаваемости.

Свойство (3.) Принадлежность графа классу Π можно определить за полиномиальное время. Другими словами класс Π полиномиально распознаваем.

Отметим, что хордальные и интервальные графы имеют полиномиальные алгоритмы распознавания [53].

Последнее свойство, которое нам потребуется, интуитивно можно описать как устойчивость относительно кликовых разделителей:

Свойство (4.) Существует алгоритм $\mathcal{A}$, принимающий на вход граф G вместе с кликой $S \subset G$. Алгоритм выдаёт «ДА» или «НЕТ» так, что выполнены следующие условия:

- Пусть $\mathcal{A}$ выдаёт «ДА» на входах (G_1, S_1) и (G_2, S_2), где $|S_1| = |S_2|$. Тогда граф G', получающийся при любом склеивании графов G_1 и G_2 так, что каждая вершина S_1 отождествляется ровно с одной вершиной из S_2, должен принадлежать классу Π.

- Если $G \in \Pi$, тогда существует разделитель-клика S такая, что $V(G) \setminus S$ можно разбить на два множества X_1, X_2, удовлетворяющих: (i) $|X_1|, |X_2| \leq \frac{2}{3}|V(G)|$, (ii) $E(X_1, X_2) = \emptyset$, (iii) алгоритм $\mathcal{A}$ отвечает «ДА» на входах $(G[X_1 \cup S], S)$ и $(G[X_2 \cup S], S)$.

Отметим, что некоторые из приведённых требований могут быть ослаблены. К примеру, свойство (3) полиномиальной распознаваемости может быть заменено на требование быть субэкспоненциально распознаваемым. Также требование быть подклассом хордальных графов может быть ослаблено в некоторой степени. Однако подобные ослабления не увеличивают число естественных классов графов, для которых применим наш алгоритм, поэтому наши требования к классу Π описаны наиболее простыми и естественными свойствами.

Заметим, что свойство (1) и предложение 2.1 гарантируют наличие $\frac{2}{3}$-сбалансированного разделителя-клики. Свойство (4), в свою очередь, позволяет проверить принадлежность классу Π, независимо проанализировав графы $G[X_1 \cup S]$ и $G[X_2 \cup S]$. Для хордальных графов, свойство (4) следует из предложения 2.1 и хорошо известного факта, что если S — разделитель-клика в графе G, а (X_1, X_2) разбиение $V(G) \setminus S$ и $E(X_1, X_2) = \emptyset$, то G — хордальный граф тогда и только тогда, когда графы $G[X_1 \cup S]$ и $G[X_2 \cup S]$ хордальные. Таким образом, если Π — это класс хордальных графов, то в качестве алгоритма $\mathcal{A}$ мы можем взять алгоритм распознавания хордальности.

Рассмотрим случай, когда Π — класс интервальных графов. Представим интервальный граф в виде пути из клик (идущего слева направо). В качестве клики S возьмём такую, что в пути слева и справа от неё находится не более $\frac{n}{2}$ вершин не из S. Пусть X_1 — это множество вершин, находящихся слева от S в пути из клик, а X_2 — множество вершин справа. Очевидно, что в таком случае S является даже $\frac{1}{2}$-сбалансированным разделителем-кликой с разбиением (X_1, X_2) множества $V(G) \setminus S$. Легко видеть, что $G[X_1 \cup S]$ и $G[X_2 \cup S]$ допускают представления в виде пути из клик, в которых последняя клика — это множество S. Покажем утверждение в обратную сторону. Если нам даны два графа G_1, G_2, допускающие представления в виде пути из клик, в которых последние клики S_1, S_2 имеют одинаковый размер, то можно создать путь из клик для графа G'. Этот путь можно положить равным непересекающемуся объединению G_1 и G_2, в котором вершины S_1 и S_2 произвольно отождествлены. Для этого достаточно просто построить путь из клик для G_1, затем рассмотреть последнюю клику S_1 как S_2 и продолжить построение пути из клик, как если бы проходило построение графа G_2. Таким образом в качестве алгоритма $\mathcal{A}$ можно взять алгоритм, принимающий на вход (G, S) и проверяющий, существует ли разложение G в путь из клик, в котором последняя клика равна S. Подобный алгоритм может быть реализован следующим образом: к графу G' добавляются две вершины v, v', где v соединено ребром с v' и всеми вершинами из S, а v'

соединено ребром только с v, после этого запускается алгоритм распознавания интервальных графов. Подобным образом добавленные вершины v, v' "заставляют" S быть последней кликой. Таким образом, мы показали, что интервальные графы тоже удовлетворяют свойству (4).

2.3 Алгоритм

В этом разделе докажем следующий главный результат этой главы:

Теорема 2.1. Если класс графов Π обладает свойствами $(1)-(4)$, тогда существует алгоритм, который принимает на вход граф G на n вершинах и возвращает максимальный индуцированный Π-подграф за время $O^*(2^{\lambda n})$ для некоторого $\lambda < 1$, где λ зависит только от Π.

Выше было доказано, что хордальные и интервальные графы удовлетворяют свойствам $(1)-(4)$. Поэтому из теоремы 2.1 немедленно следует существование алгоритмов для нахождения наибольших хордальных и интервальных подграфов за время $2^{\lambda n}$. Наш подход основан на тщательном изучении структуры максимального индуцированного подграфа. Для уменьшения числа потенциальных решений будут применены значительно отличающиеся между собой методы, основанные на информации о структуре решения. В нашем алгоритме мы сильно полагаемся на наличие сбалансированного разделителя (свойство (4)) и условия на минимальные запрещённые подграфы класса Π (свойство (2)). Алгоритм схематично изображён на рис. 2.2.

Пусть $G = (V, E)$. На протяжении описания алгоритма мы будем использовать несколько малых положительных константных величин: $\alpha, \beta, \gamma, \delta, \epsilon$ и одну "большую" константу L (краткое описание констант представлено в таблице 2.1). Значение константы λ зависит от выбора значений констант $\alpha, \beta, L, \gamma, \delta, \epsilon$. В разделе 2.4 мы покажем, что константы $(\alpha, \beta, L, \gamma, \delta, \epsilon)$ могут быть выбраны так, что $\lambda < 1$. Выбор значения каждой константы зависит от значения последующих констант. То есть, выбрав значения для констант $L, \gamma, \delta, \epsilon$, можно найти верхнюю оценку

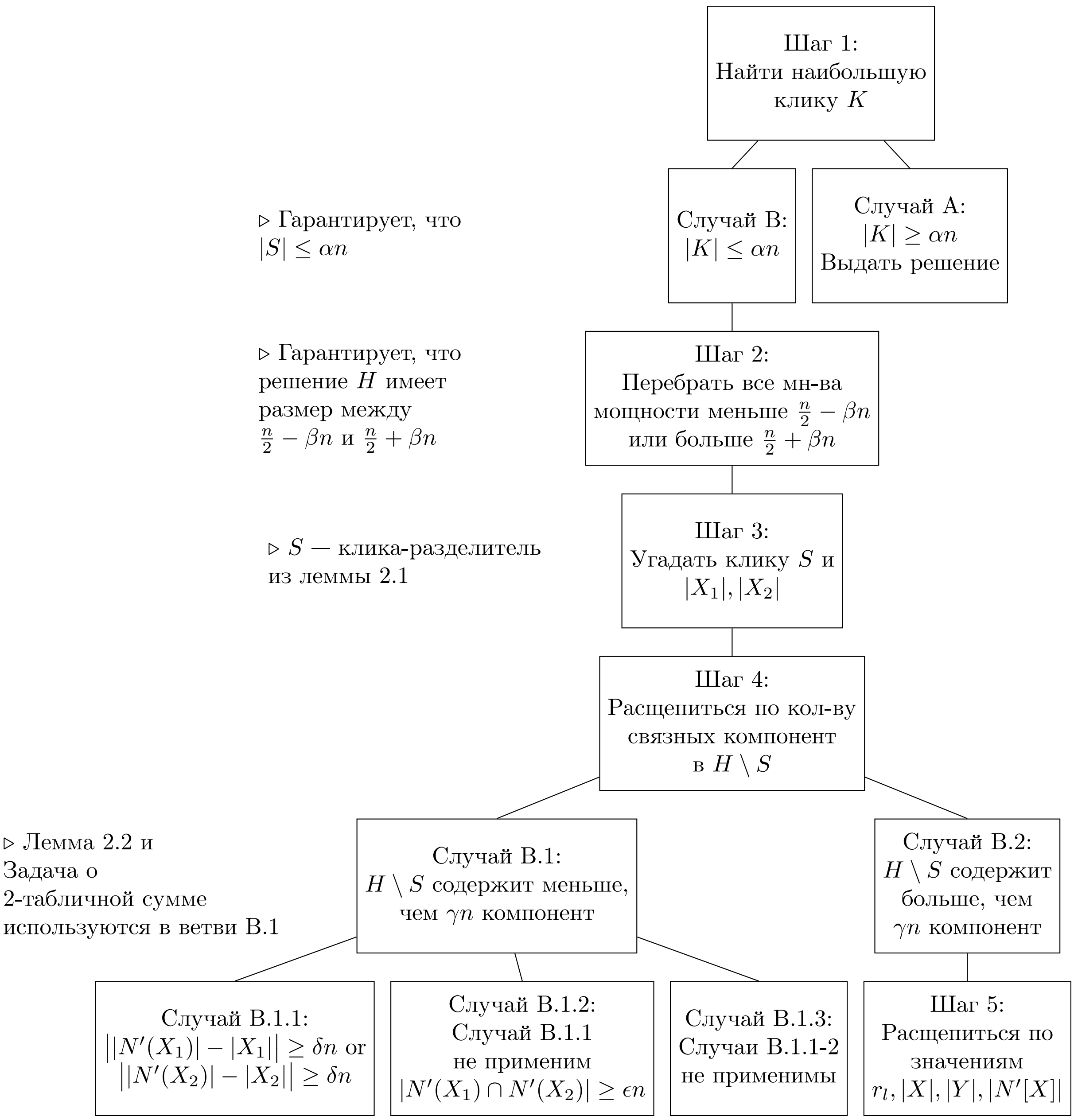


Рис. 2.2: Схематичное представление работы алгоритма.

на β так, что в качестве значения β можно взять любое число, меньшее полученной оценки.

Заметим, что, благодаря лемме 2.1, можно предположить, что граф G, поданный на вход, не содержит запрещённых индуцированных подграфов из множества $\mathcal{F}_\Pi$, содержащих менее $\ell + 1$ вершин (значение ℓ будет выбрано позже). Действительно, если у нас есть алгоритм для поиска наибольшего индуцированного П-подграфа, требующего $O^*(2^{\lambda n})$ времени для $\lambda < 1$, и работающего на $\mathcal{F}'$-свободных графах (где $\mathcal{F}'$ — подмножество всех графов из $\mathcal{F}_\Pi$, содержащих более ℓ вершин), тогда по лемме 2.1 мы получим алгоритм поиска наибольшего индуцированного П-подграфа, работающего на всех входах и требующего $2^{\lambda' n}$ времени, где $\lambda' < 1$. С этого момента будем считать, что входной граф G не содержит подграфов из множества $\mathcal{F}_\Pi$ с числом вершин менее $\ell + 1$.

Наш алгоритм состоит из нескольких этапов (алгоритм схематично представлен на рис. 2.2).

Шаг 1. *Используя алгоритм Робсона [10], за время $\mathcal{O}^*(2^{0.276n})$ найдём наибольшую клику K в графе G.*

Мы рассмотрим два случая: (a) клика K достаточно большая и мы можем закончить поиск, применив специальный алгоритм, или (b) клика K мала, и у нас есть гарантия, что наибольший индуцированный П-граф, который мы ищем, содержит только небольшие клики. Пороговое значение на размер клики (большая клика или нет) равно αn для некоторой константы α, $0 < \alpha < 1/48$ (выбор значения константы будет описан позднее).

Случай A: $|K| \geq \alpha n$.

Мы покажем, что в этом случае задача может быть решена за $O^*(2^{(1-(1-\kappa_0)\alpha)n})$ времени для некоторого $\kappa_0 < 1$, зависящего от $\aleph$. Воспользуемся следующей вспомогательной леммой:

Лемма 2.4. Пусть P — это П-подграф графа G, а $K \subset G$ — это клика, такая что $P \cap K = \emptyset$. Тогда максимальный индуцированный подграф $H \subset G$, удовлетворяющий

- (i) $H \in \Pi$,
- (ii) $V(H) \setminus K = P$,

может быть найден за время $\mathcal{O}^*(2^{\kappa_0|K|})$.

Доказательство. Для удобства раскрасим некоторые подмножества вершин клики K. Покрасим $W \subset K$ в красный цвет, если $W \neq \emptyset$, $|W| \leq \aleph$ и $G[W \cup P] \in P$. Заметим, что подобная покраска может быть произведена с использованием не более $\aleph \cdot |K|^\aleph$ проверок на принадлежность классу Π, а значит за полиномиальное время для константного значения $\aleph$.

Заметим, что для любого $X \subset K$ граф $G[P \cup X]$ принадлежит классу Π тогда и только тогда, когда все непустые подмножества X мощности не больше $\aleph$ покрашены в красный цвет. Докажем этот факт. Предположим, не все такие подмножества покрашены в красный цвет, тогда существует $W \subset X$ такое, что $G[W \cup P] \notin \Pi$, а значит по свойству (1) получаем $G[P \cup X] \notin \Pi$.

Для доказательства в обратную сторону предположим, что $G[P \cup X]$ содержит какой-то запрещённый индуцированный подграф $F \in \mathcal{F}_\Pi$. Тогда $|F \cap X| > \aleph$, иначе, по построению раскраски $F \cap X$ не было бы покрашено в красный цвет. Но поскольку X — это клика, то получается, что F содержит клику с $\aleph + 1$ вершиной, что противоречит свойству (2).

Таким образом, чтобы найти требуемый подграф, необходимо найти наибольшее подмножество X такое, что все его подмножества, состоящие не более, чем из $\aleph$ элементов, покрашены в красный цвет. Таким образом, мы свели задачу к поиску максимальной клики в гиперграфе, у которого все гиперрёбра имеют размер не больше $\aleph$. Подобную клику можно легко найти с помощью расщепляющегося алгоритма за время $2^{\kappa_0 \cdot |K|}$, где $\kappa_0 < 1$ и зависит только от значения $\aleph$.

Теперь опишем подробнее работу расщепляющегося алгоритма для поиска клики в гиперграфе. Алгоритм постоянно хранит два непересекающихся множества вершин A и D. Изначально $A = D = \emptyset$. Множество A содержит вершины, которые предположительно принадлежат ответу, в то время как множество D содержит вершины, не принадлежащие решению. Вычисление ветки алгоритма останавливается в тот момент, когда все подмножества множества $K \setminus D$, состоящие из не более $\aleph$ элементов, покрашены в красный цвет. Такое множество $K \setminus D$ будет считаться кандидатом на решение. Решением X будет наибольшее множество из всех таких кандидатов (максимум по всем листьям). Если вычисления в ветке алгоритма не остановлены, значит существует подмножество $W \subset K \setminus D$, не покрашенное в красный цвет и $|W| \leq \aleph$. Очевидно, что как минимум одна вершина из W не принадлежит решению. Поэтому мы можем рассмотреть $W \setminus A$ и расщепиться на $2^{|W \setminus A|} - 1$ случай, в каждом случае определяя новое подмножество $W \setminus A$, которое будет включено в A. Оставшиеся вершины $W \setminus A$ в этой ветке расщепления будем считать не принадлежащими решению и поэтому добавим их к множеству D. Здесь полного перебора удаётся избежать, поскольку заведомо известно, что все вершины $W \setminus A$ в текущей ветке расщепления не могут принадлежать решению. Поскольку $|W \setminus A| \leq \aleph$, то $2^{|W \setminus A|} - 1 \leq 2^{\kappa_0 |W \setminus A|}$, где $\kappa_0 < 1$ зависит только от $\aleph$. Таким образом нам удалось рассмотреть все случаи распределения вершин $W \setminus A$, создав не более $2^{\kappa_0 \cdot |W \setminus A|}$ ветвей расщепления. Тем самым, общее время работы алгоритма не превосходит $2^{\kappa_0 \cdot |K|}$. $\square$

Пусть H — это искомый ответ, т.е. наибольший индуцированный подграф G из класса Π. Для завершения рассмотрения случая A, нам достаточно перебрать $2^{|V \setminus K|}$ различных возможностей для множества P. Напомним, что P — это подмножество $V \setminus K$, и оно играет роль $V(H) \setminus K$; мы отбросим случаи, в которых $P \notin \Pi$. Для каждого оставшегося случая мы применим лемму 2.4 и найдём наибольший Π-подграф, содержащий вершины P. В каждом таком случае мы затратим не более $\mathcal{O}^*(2^{\kappa_0 \cdot |K|})$ времени. Тем самым, мы получаем, что время работы всего алгоритма составит не более $\mathcal{O}^*(2^{|V \setminus K|} \cdot 2^{\kappa_0 \cdot |K|}) \leq \mathcal{O}^*(2^{(1-\alpha)n} \cdot 2^{\kappa_0 \cdot \alpha n}) = \mathcal{O}^*(2^{(1-(1-\kappa_0)\alpha)n})$. Заметим, что $1 - (1 - \kappa_0)\alpha < 1$ для $\alpha > 0$ и $\kappa_0 < 1$.

Случай B: *Граф G не содержит клики размера αn.*

Сперва покажем, как найти решение, если его размер значительно отличается от $\frac{n}{2}$. Для этого мы просто переберём все подмножества с размером больше $\frac{n}{2} + \beta n$ и все подмножества с размером меньше $\frac{n}{2} - \beta n$, где β — некоторая константа из интервала $[0, \frac{1}{16}]$. Заметим, что по лемме 2.2 $\mathcal{O}^*(\binom{n}{\lceil n/2-\beta n \rceil}) \le \mathcal{O}^*(2^{\kappa_1 n})$ для некоторого $\kappa_1 < 1$, зависящего от β.

Шаг 2. Для каждого подграфа с количеством вершин, большим $\frac{n}{2} + \beta n$ или меньшим $\frac{n}{2} - \beta n$, проверим его принадлежность классу П. Пусть H будет наибольшим таким подграфом из класса П. Если $|H| \ge \frac{n}{2} + \beta n$, то в качестве ответа выдадим H и остановим алгоритм. Если $|H| = \frac{n}{2} - \beta n$, то продолжим поиски решения среди подграфов, чей размер находится в промежутке между $\frac{n}{2} - \beta n$ и $\frac{n}{2} + \beta n$. Если $|H| < \frac{n}{2} - \beta n$, то в качестве решения выдадим H и остановим алгоритм. Данная процедура корректна, поскольку класс П обладает свойством наследственности, а значит существование П-подграфа с более $\frac{n}{2} - \beta n$ вершинами влечёт также существование П-подграфа ровно с $\frac{n}{2} - \beta n$ вершинами.

Если после выполнения шага 2 алгоритм не остановился, мы знаем, что размер наибольшего индуцированного П-подграфа находится в интервале $[\frac{n}{2} - \beta n, \frac{n}{2} + \beta n]$. Будем использовать данный факт в последующих шагах алгоритма.

Пусть H — наибольший индуцированный П-подграф G. На данный момент мы знаем следующее о подграфе H:

- H не содержит клики размера αn,
- $n/2 - \beta n \le |V(H)| \le n/2 + \beta n$.

Заметим, что количество вершин в графе G, не принадлежащих H, тоже находится в промежутке между $n/2 - \beta n$ и $n/2 + \beta n$.

Воспользуемся свойством (4), чтобы найти $\frac{2}{3}$-сбалансированный разделитель-клику в подграфе H. Мы знаем, что существует клика $S \in H$, такая что множество $V(H) \setminus S$ можно разбить на два подмножества X_1, X_2 и выполнено следующее:

- $\frac{1}{3}|V(H)| - |S| \le |X_1|, |X_2| \le \frac{2}{3}|V(H)|$,
- $E(X_1, X_2) = \emptyset$.

Поскольку S — это клика в графе G, то $|S| \le \alpha n$. Так как $\beta < 1/16$ и $\alpha < 1/48$, получается, что $|X_1|, |X_2| \ge (\frac{1}{6} - \frac{\beta}{3} - \alpha)n > \frac{1}{8}n$. Поскольку α — малая величина, мы можем произвести следующее расщепление на подзадачи:

Шаг 3. *Создадим не более $(1 + \alpha n)\binom{n}{\alpha n} \cdot (n+1)^2$ подзадач, в каждой зафиксировав подмножество $S \subset V$, размером не больше αn, также зафиксируем размеры подмножеств X_1, X_2 (ограничим $\binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{\alpha n}$ с помощью $(1 + \alpha n) \cdot \binom{n}{\alpha n}$). Сразу удалим те подзадачи, в которых зафиксированное подмножество S не является кликой.*

Далее сосредоточим внимание на решении одной подзадачи. То есть мы считаем, что множество S и значения $|X_1|, |X_2|$ в дальнейшем известны. Пусть $G' = G \setminus S$; для удобства обозначений при $X \subset V(G')$ положим $N'[X] = N_{G'}[X]$ и $N'(X) = N_{G'}(X)$. В зависимости от количества компонент связности в графе $H \setminus S$ рассмотрим два случая: не более γn компонент и более γn, где γ — некоторая константа, большая нуля. Значение γ будет выбрано позже.

Шаг 4. *Создадим два новых подслучая, в первом предположим, что $H \setminus S$ содержит не больше γn связных компонент, а во втором будем считать, что $H \setminus S$ состоит из более, чем γn компонент связности.*

В каждом из подслучаев будем рассматривать новые расщепления и подзадачи, но для упрощения описания не будем разделять их на отдельные шаги, как это было сделано ранее.

Случай B.1: *Граф $H \setminus S$ содержит не более γn связных компонент.* Расщепимся (создадим) на $(n+1)^3$ случаев, в каждом зафиксировав мощности множеств $N'(X_1), N'(X_2)$ и $N'(X_1) \cap N'(X_2)$ так, что выполнены неравенства $|N'(X_1) \cap N'(X_2)| \le |N'(X_1)|, |N'(X_2)| \le n - (|S| + |X_1| + |X_2|)$. С этого момента считаем, что значения

$|N'(X_1)|, |N'(X_2)|, |N'(X_1) \cap N'(X_2)|$ известны. Далее в рассматриваемых подслучаях будем использовать малые константы δ и ε, значения которых будут подобраны позже.

Случай B.1.1: $\big||N'(X_1)| - |X_1|\big| \geq \delta n$ *или* $\big||N'(X_2)| - |X_2|\big| \geq \delta n$.

Рассмотрим только случай $\big||N'(X_1)| - |X_1|\big| \geq \delta n$, поскольку другой случай симметричен. Известно, что количество компонент в $H \setminus S$ невелико, поэтому их расположение можно перебрать с помощью расщепления на несколько подзадач. Представим себе множество P_1, получающееся, если из каждой связной компоненты $G[X_1] = H[X_1]$ выбрать одну вершину. Мы знаем, что $|P_1| \leq \gamma n$. Создадим не более, чем $(1+\gamma n)\binom{n}{\gamma n}$ подзадач, в каждой из подзадач зафиксировав в качестве P_1 подмножество из не более, чем γn вершин.

Добавим искусственную вершину v_1 к графу G' и соединим её со всеми вершинами из P_1. В полученном графе пронумеруем все связные подграфы размера $|X_1|+1$, содержащие множество вершин $\{v_1\} \cup P_1$ и имеющие окрестность из $N'(X_1)$ вершин. По лемме 2.2 количество таких подграфов не превосходит $\binom{|X_1|+|N'(X_1)|}{|X_1|}$ и они могут быть перечислены за время $O^*(\binom{|X_1|+|N'(X_1)|}{|X_1|})$; отметим, что мы перечисляем такие множества, пользуясь леммой 2.2 для вершины v_1, а после этого удаляем все множества, не содержащие $\{v_1\} \cup P_1$ в качестве подмножества. Очевидно, множество $X_1 \cup \{v_1\}$ будет в списке перечисленных.

Таким образом, мы создали не более $\binom{|X_1|+|N'(X_1)|}{|X_1|}$ подзадач, в каждой зафиксировав некоторое множество в качестве X_1 (после того, как удалили искусственную вершину v_1). В каждой из этих подзадач мы рассмотрим $\binom{n-|X_1|-|N'(X_1)|}{|X_2|}$ вариантов для множества X_2. Напомним, что $X_2 \subset V \setminus (N'[X_1] \cup S)$. Для каждого такого варианта за полиномиальное время проверим, принадлежит ли подграф $G[X_1 \cup X_2 \cup S]$ классу Π .

Таким образом, общее количество всех вариантов во всех подслучаях случая $B.1.1$ не превосходит

$$(1+\gamma n) \cdot \binom{n}{\gamma n} \cdot \binom{|X_1| + |N'(X_1)|}{|X_1|} \cdot \binom{n - |X_1| - |N'(X_1)|}{|X_2|}.$$

Поскольку $|N'(X_1)|, |X_1| \le n$ и $\big||N'(X_1)| - |X_1|\big| \ge \delta n$, то по лемме 2.2 получим $\binom{|X_1|+|N'(X_1)|}{|X_1|} \le \mathcal{O}^*(2^{\kappa_2(|X_1|+|N'(X_1)|)})$ для некоторого $\kappa_2 < 1$, зависящего от δ. В свою очередь,

$$\binom{n - |X_1| - |N'(X_1)|}{|X_2|} \le \mathcal{O}^*(2^{n-|X_1|-|N'(X_1)|}).$$

Ранее было показано (конец шага 2), что $|X_1| \ge \frac{1}{8}n$, а значит и $|N'(X_1)| + |X_1| \ge \frac{1}{8}n$. Таким образом, мы имеем:

$$\binom{|X_1| + |N'(X_1)|}{|X_1|} \cdot \binom{n - |X_1| - |N'(X_1)|}{|X_2|} = \mathcal{O}^*(2^{\kappa_3 n}),$$

где константа $\kappa_3 < 1$ зависит от δ. Учитывая все расщепления, полученные во время рассмотрения случая B.1.1, включая перебор кандидатов на роль S и фиксирование мощностей множеств, получаем, что затраченное время ограничено сверху $\mathcal{O}^*(\binom{n}{\alpha n} \cdot \binom{n}{\gamma n} \cdot 2^{\kappa_3 n})$. Поскольку κ_3 зависит только от δ, то при фиксированном δ можно выбрать α, γ достаточно малыми так, что время не превосходит $\mathcal{O}^*(2^{\kappa_4 n})$, где $\kappa_4 < 1$.

Случай B.1.2: *Случай B.1.1 не применим, но* $|N'(X_1) \cap N'(X_2)| \ge \varepsilon n$.

Будем действовать аналогично случаю B.1.1, но изменим стратегию перебора множества X_2: вместо перебора всех подмножеств $V \setminus (N'[X_1 \cup S])$, воспользуемся леммой 2.2. Пусть множества P_1, P_2 получаются выбором по одной вершине из каждой компоненты связности в графах $G[X_1] = H[X_1]$ и $G[X_2] = H[X_2]$ соответственно. Напомним, что $|P_1|, |P_2| \le \gamma n$. Аналогично предыдущему случаю создадим $(1 + \gamma n)^2 \cdot \binom{n}{\gamma n}^2$ подзадач, в каждой зафиксировав подмножества P_1 и P_2. Как и ранее, добавим искусственную вершину и соединим её со всеми вершинами из P_1. Аналогично случаю B.1.1 перечислим не более $\binom{|X_1|+|N'(X_1)|}{|X_1|}$ кандидатов на роль X_1. После этого для каждого подобного кандидата X_1 создадим свою подзадачу. Не будем рассматривать те подзадачи, в которых P_2 и X_1 пересекаются или множество $E(P_2, X_1)$ не пусто. Заметим, что на данный момент мы создали не более $\binom{|X_1|+|N'(X_1)|}{|X_1|} \le \mathcal{O}^*(2^{2|X_1|+\delta n})$ подзадач.

Рассмотрим граф $G'' = G \setminus (N'[X_1] \cup S)$. Заметим, что $X_2 \subseteq V(G'')$ и мощность окрестности X_2 в G'' не превосходит $|N'(X_2)| - \varepsilon n$, поскольку уделены как минимум εn вершин, лежащих в пересечении $N'(X_1)$ и $N'(X_2)$. Теперь добавим в G'' искусственную вершину v_2 и соединим её со всеми вершинами из P_2, после чего применим лемму 2.2. Аналогично, как и с X_1, перечислим не более

$$\binom{|X_2| + |N'(X_2)| - |N'(X_1) \cap N'(X_2)|}{|X_2|}$$

подмножеств в качестве кандидатов на роль X_2. При этом затратим не более

$$\mathcal{O}^*\left(\binom{|X_2| + |N'(X_2)| - |N'(X_1) \cap N'(X_2)|}{|X_2|}\right)$$

времени. Таким образом, мы создадим не более $\binom{|X_2|+|N'(X_2)|-|N'(X_1)\cap N'(X_2)|}{|X_2|}$ подзадач, в каждой зафиксировав кандидата на роль X_2. Поскольку $|N'(X_2)| \leq |X_2| + \delta n$ и $|N'(X_1) \cap N'(X_2)| \geq \varepsilon n$, получаем $|X_2| + |N'(X_2)| - |N'(X_1) \cap N'(X_2)| \leq 2|X_2| - (\varepsilon - \delta)n$.

В каждой из подзадач множества X_1 и X_2 зафиксированы. За полиномиальное время проверим, верно ли, что $G[X_1 \cup X_2 \cup S] \in \Pi$. Заметим, что $\mathcal{O}^*(2^{2|X_1|+\delta n}) \cdot \mathcal{O}^*(2^{2|X_2|-(\varepsilon-\delta)n}) = \mathcal{O}^*(2^{2(|X_1|+|X_2|)-(\varepsilon-2\delta)n})$; более того, $|X_1| + |X_2| \leq n/2 + \beta n$. Таким образом, при фиксированном $\varepsilon > 0$ можно выбрать значения δ и β достаточно малыми так, что $\mathcal{O}^*(2^{2|X_1|+\delta n}) \cdot \mathcal{O}^*(2^{2|X_2|-(\varepsilon-\delta)n}) \leq \mathcal{O}^*(2^{\kappa_5 n})$ для некоторого $\kappa_5 < 1$, зависящего от ε.

Подводя итог, получаем, что общее количество подслучаев, сгенерированных в случае B.1.2, учитывая перебор S, мощностей множеств и P_1, P_2, не превосходит $\mathcal{O}^*(\binom{n}{\alpha n} \cdot \binom{n}{\gamma n}^2) \cdot \mathcal{O}^*(2^{\kappa_5 n})$. При фиксированном значении κ_5 можно выбрать достаточно малые значения α и γ так, что общее количество подзадач не превосходит $\mathcal{O}^*(2^{\kappa_6})$, где $\kappa_6 < 1$. Общее затраченное время также составит $\mathcal{O}^*(2^{\kappa_6})$, так как каждая подзадача требует лишь полиномиального времени.

Случай B.1.3: *Ни один из случае B.1.1 или B.1.2 не применим.*

Напомним, что к этому моменту множества X_1, X_2 удовлетворяют следующему:

- $\frac{1}{6}n - \frac{\beta}{3}n - \alpha n \leq |X_1|, |X_2| \leq \frac{1}{3}n + \frac{2\beta}{3}n$,
- $\frac{1}{2}n - (\alpha+\beta)n \leq |X_1| + |X_2| \leq \frac{1}{2}n + \beta n$,
- $\big||N'(X_i)| - |X_i|\big| \leq \delta n$ для $i = 1, 2$, и $|N'[X_1] \cap N'[X_2]| \leq \varepsilon n$.

Пусть $U_{\text{both}} = N'[X_1] \cap N'[X_2] = N'(X_1) \cap N'(X_2)$, $U_{\text{none}} = V(G') \setminus (N'[X_1] \cup N'[X_2])$ и $U = U_{\text{both}} \cup U_{\text{none}}$. Мы знаем, что $|U_{\text{both}}| \leq \varepsilon n$. Покажем, что $|U_{\text{none}}| \leq \zeta n$, где $\zeta = 2\alpha + 2\beta + 2\delta + \varepsilon$:

$$\begin{aligned} |U_{\text{none}}| &= |V(G')| - |X_1| - |X_2| - |N'(X_1)| - |N'(X_2)| + |N'(X_1) \cap N'(X_2)| \\ &\leq n - 2(|X_1| + |X_2|) + 2\delta n + \varepsilon n \leq (2\alpha + 2\beta + 2\delta + \varepsilon)n. \end{aligned}$$

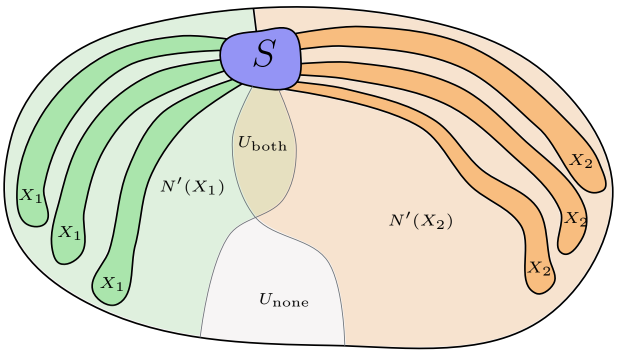


Рис. 2.3: Структура случая B.1.3. Размеры окрестностей $N'(X_1), N'(X_2)$ почти равны мощностям X_1, X_2 соответственно, в то время как U_{both} и U_{none} содержат лишь малую часть общего количества вершин.

Учитывая, что размеры множеств U_{both} и U_{none} малы, мы можем перебрать их возможные значения, увеличив время работы не более чем в $\mathcal{O}^*(\binom{n}{\varepsilon n} \cdot \binom{n}{\zeta n})$ раз. Для этого создадим $\mathcal{O}^*(\binom{n}{\varepsilon n} \cdot \binom{n}{\zeta n})$ подзадач, в каждой из подзадач фиксируя подмножества $U_{\text{both}}, U_{\text{none}} \subset V \setminus S$, учитывая, что $U_{\text{both}} \leq \varepsilon n$ и $U_{\text{none}} \leq \zeta n$. Заметим, что $V(G') \setminus U$ равняется симметрической разности $N'[X_1]$ и $N'[X_2]$. Пусть $I = V(G') \setminus U$. Теперь осталось определить, какие вершины I принадлежат $X_1 \cup X_2$, а какие нет.

Заметим, что $(N[X_1], N[X_2])$ является разбиением множества вершин I. А значит, из свойства (4) следует, что нам достаточно искать такие $X_1, X_2 \subset I$, что (i) алгоритм $\mathcal{A}$ выдаёт ответ «ДА» на входах $G[X_1 \cup S], G[X_2 \cup S]$ с выделенной кликой S, и (ii) I представимо в виде разбиения $N[X_1] \sqcup N[X_2]$. Заметим, что данную ситуацию можно смоделировать с помощью задачи о 2-Таблице. Для $i = 1, 2$ мы перечислим все подмножества I размера $|X_i|$, как кандидатов на роль X_i. Удалим множество X из списка кандидатов, если алгоритм $\mathcal{A}$ не выдаёт ответ «ДА» на графе $X \cup S$. Для каждого оставшегося множества-кандидата создадим бинарный вектор длины $|I|$, показывающий, какие вершины I принадлежат множеству-кандидату или его окрестности (значение 1 в соответствующей координате вектора), а какие нет (значение 0). Используя векторы множеств-кандидатов как столбцы, построим матрицы T_1, T_2 для множеств X_1, X_2 соответственно. Теперь для решения задачи достаточно найти столбец в T_1 и столбец в T_2 такие, что их сумма равняется столбцу, состоящему из одних единиц.

Матрицы T_1, T_2 содержат не более $\binom{n}{\frac{1}{3}n+\frac{2\beta}{3}n}$ столбцов, поскольку $|X_1|, |X_2| \le \frac{1}{3}n + \frac{2\beta}{3}n$. То есть количество столбцов не больше $\mathcal{O}^*(2^{\kappa_7 n})$, для некоторого $\kappa_7 < 1$, ведь $\beta < 1/16$ (а значит $\frac{1}{3} + \frac{2\beta}{3}n \le \frac{3}{8}$). Следовательно, по лемме 2.3 можно решить полученную задачу о 2-табличной сумме за время $\mathcal{O}^*(2^{\kappa_7 n})$. Тем самым мы получаем, что общее количество времени, также затраченное на перебор множеств S, U_{both}, U_{none} и мощностей множеств не превосходит $\mathcal{O}^*(\binom{n}{\alpha n} \cdot \binom{n}{\varepsilon n} \cdot \binom{n}{\zeta n} \cdot 2^{\kappa_7 n})$. Отметим, что значения $\alpha, \beta, \delta, \varepsilon$ могут быть выбраны достаточно малыми так, чтобы всё затраченное время составило не более $\mathcal{O}^*(2^{\kappa_8 n})$ для некоторого $\kappa_8 < 1$.

Случай B.2: *Граф $H \setminus S$ содержит больше γn связных компонент.*

Рассмотрим связные компоненты графа $H \setminus S$ и зафиксируем большую константу $L > 2$, зависящую от γ (как выбрать значение константы будет указано позже). Будем называть связную компоненту *малой*, если она содержит не более $C = L/\gamma$ вершин, в противном случае компоненту будем называть *большой*. Пусть r_ℓ и r_s обозначают количество больших и малых компонент в $H \setminus S$ соответственно. Число вершин, содержащихся в больших компонентах, не менее $\frac{L \cdot r_\ell}{\gamma}$. Таким образом, $\frac{L \cdot r_\ell}{\gamma} \leq n$, $r_\ell \leq \frac{\gamma n}{L}$ и следовательно, $r_s \geq \gamma n - r_\ell \geq \gamma n(1 - \frac{1}{L}) \geq \frac{\gamma n}{2}$. Поскольку любая малая компонента содержит как минимум одну вершину, то суммарно в малых компонентах содержится не менее $\frac{\gamma n}{2}$ вершин.

Кратко опишем структуру задачи, которую мы имеем на данный момент (см. рис. 2.4). Вершины V могут быть разделены на попарно непересекающиеся множества S, X, N_X, Y, Z, где:

(i) S — клика, выбранная на шаге 3;

(ii) X — вершины, содержащиеся в больших компонентах графа $H \setminus S$;

(iii) $N_X = N'(X)$;

(iv) Y — вершины, содержащиеся в малых компонентах $H \setminus S$;

(v) Z состоит из вершин, не содержащихся в H и не соединённых с X.

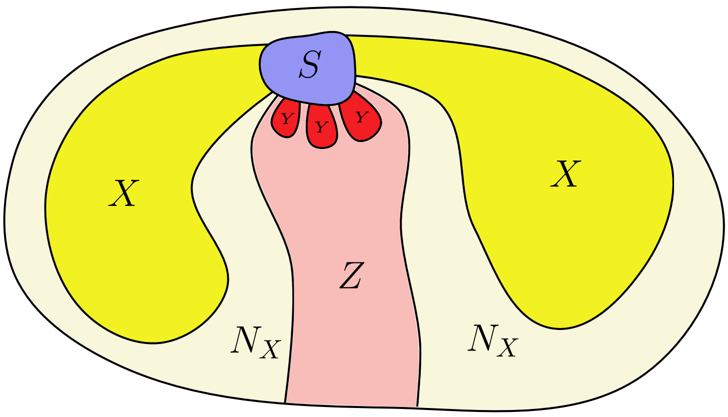


Рис. 2.4: Структура в случае B.2. Заметим, что даже зафиксировав X и S, необходимо разбить оставшуюся часть $V \setminus (N'[X] \cup S)$ на множества Y и Z.

Заметим, что $V(H) = S \cup X \cup Y$. К сожалению, даже если известны X и S, алгоритм не может немедленно получить решение: мы обязаны разделить оставшиеся вершины $V \setminus (N'[X] \cup S)$ на множество Y (вершины из искомого графа H) и Z. Однако известно, что $G[X]$ содержит малое количество компонент, поэтому мы можем перебрать кандидатов на роль множества X, используя лемму 2.2. Пусть множество P содержит ровно по одной вершине из каждой компоненты связности графа $G[X]$. Известно, что $|P| = r_\ell \leq \frac{\gamma n}{L}$.

Шаг 5. *Рассмотрим* $(n+1)^4$ *различных вариантов значений* $r_\ell, |X|, |Y|, |N'[X]|$. *После этого переберём* $\binom{n}{r_\ell} \leq \binom{n}{\frac{\gamma n}{L}}$ *случаев, в каждом выбрав соответствующего кандидата на роль множества* P. *Добавим искусственную вершину* v_1 *и соединим её с* P. *Используя лемму 2.2, за время* $\mathcal{O}^*(\binom{|N'[X]|}{|X|}) \leq \mathcal{O}^*(2^{|N'[X]|})$ *перечислим не более* $\binom{|N'[X]|}{|X|} \leq 2^{|N'[X]|}$ *подграфов, которые (i) связны, (ii) содержат* $P \cup \{v_1\}$, *(iii) содержат* $|X|+1$ *вершину и мощность их окрестности равна* $|N'(X)|$. *Отметим, что данную процедуру можно проделать, просто отфильтровав список множеств, выданных леммой 2.2, удалив множества, не содержащие* P *в качестве подмножества. Отметим, что среди перечисленных множеств находится* $X \cup v$.

Пусть $R = G[V \setminus (N'[X] \cup S)]$. Должны выполняться неравенства $|V(R)| \geq |Y| \geq r_s \geq \frac{\gamma n}{2}$, а значит, если $|V(R)| < \frac{\gamma n}{2}$, то завершаем работу алгоритма в данном подслучае. В последующей лемме воспользуемся тем свойством, что граф, поданный на вход, не содержит запрещённых индуцированных подграфов с количеством вершин, не превосходящих некоторой константы ℓ. Напомним, что это предположение было обосновано в начале текущего раздела по лемме 2.1. Положим $\ell = 3C^2 + 1$. Таким образом, любой индуцированный подграф G, имеющий не более ℓ вершин, будет принадлежать классу Π. Последующие шаги алгоритма заключены в следующей лемме.

Лемма 2.5. Положим $\alpha < \frac{\gamma}{104C^3}$ и $\ell = 3C^2 + 1$. Существует универсальная константа $\rho < 1$ и алгоритм, который перечисляет не более $\mathcal{O}(2^{\rho|V(R)|})$ подмножеств $V(R)$ так, что множество Y находится среди перечисленных подмножеств и время работы алгоритма не превосходит

$\mathcal{O}^*(2^{\rho|V(R)|})$.

Прежде чем приступить к доказательству леммы, покажем, что применение леммы завершает работу алгоритма. Действительно, на данный момент мы создали не более $\mathcal{O}^*(\binom{n}{\alpha n} \cdot \binom{n}{\frac{\gamma n}{L}} \cdot 2^{|N'[X]|})$ подзадач, перебирая все возможные подмножества в качестве S и X. Если мы сейчас перечислим всех кандидатов на роль Y по лемме 2.5, а потом запустим тест на принадлежность графа $G[X \cup Y \cup S]$ классу Π, то использованное время составит:

$$\mathcal{O}^*\left(\binom{n}{\alpha n} \cdot \binom{n}{\frac{\gamma n}{L}} \cdot 2^{|N'[X]|} \cdot 2^{\rho|V(R)|}\right).$$

Поскольку $|N'[X]| + |V(R)| \leq n$, $\rho < 1$ — универсальная константа и $|V(R)| \geq \frac{\gamma n}{2}$, то при фиксированном $\gamma > 0$ можно взять L достаточно большим, а $\alpha > 0$ достаточно малым (в частности, меньше $\frac{\gamma}{104C^3}$) так, что время работы составит $\mathcal{O}^*(2^{\kappa_9 n})$ для некоторого $\kappa_9 < 1$. Здесь мы пользуемся тем, что ρ не зависит от α, γ и L. Отметим, что пороговое значение для C зависит от γ и L. А значит, и пороговое значение ℓ на размер запрещённых индуцированных графов, по которым мы расщепились, используя лемму 2.1, зависит от γ и L. Однако данное расщепление произошло до описания текущего алгоритма, что позволяет избавиться от циклической зависимости в определении значений констант.

Приступим к доказательству леммы 2.5.

Доказательство. Вначале применим классический расщепляющий алгоритм, цель которого — "уменьшить" степени вершин в R. Алгоритм создаст несколько подзадач. Каждой подзадаче будет соответствовать пара (A, D), где множество A состоит из вершин, предположительно принадлежащих решению (множеству Y), а множество D — из вершин, не принадлежащих решению. На протяжении работы расщепляющего алгоритма будем добавлять вершины в множества A и D. Первоначально $A = D = \emptyset$. Будем называть вершину $v \in V(R)$ *тяжёлой*, если $deg_{R \setminus D}(v) \geq 3C$, и *лёгкой* в противном случае. Цель такого подалгоритма — избавиться от тяжёлых вершин, то есть достичь такой ситуации, в которой все вершины в $R \setminus D$ лёгкие (напомним, что степени вершин берутся относительно графа $R \setminus D$). Следующее утверждение формально описывает ситуацию, которую мы хотим достичь:

Утверждение 2.2. Существует константа $\sigma < 1$ и алгоритм с временем работы $\mathcal{O}^*(2^{\sigma|V(R)|})$, который выдает множество пар $\mathcal{L} = \{(A_1, D_1), (A_2, D_2), \ldots, (A_p, D_p)\}$ непересекающихся подмножеств $V(R)$ так, что выполнены следующие условия:

- Для любого i все вершины множества $R \setminus D_i$ лёгкие.
- Существует индекс i_0 такой, что $A_{i_0} \subseteq Y$ и $D_{i_0} \cap Y = \emptyset$.
- $\sum_{i=1}^{p} \phi(A_i, D_i) \le 2^{\sigma|V(R)|}$, где ϕ — потенциальная функция, определённая следующим образом: $\phi(A, D) = 2^{\sigma|V(R) \setminus (A \cup D)|}$.

Доказательство утверждения 2.2. Напомним, что алгоритм постоянно хранит два непересекающихся множества A и D; A содержит вершины, предположительно содержащиеся в решении, D — вершины вне решения. Первоначально $A = D = \emptyset$. Алгоритм перестаёт генерировать подслучаи, как только множество $R \setminus D$ перестаёт содержать тяжёлые вершины.

Прогресс алгоритма измеряется потенциальной функцией $\phi(A, D) = 2^{\sigma|V(R) \setminus (A \cup D)|}$, где $\sigma < 1$ — некоторая константа, значение которой будет выбрано позже. При каждом расщеплении, мы гарантируем, что сумма потенциалов подзадач не превосходит значения потенциала первоначальной задачи (до расщепления). Поскольку значение потенциала не меньше 1, то всего будет выдано не более $2^{\sigma|V(R)|}$ пар (A_i, D_i), и их суммарный потенциал будет не более $2^{\sigma|V(R)|}$. Поскольку каждое расщепление осуществляется за полиномиальное время, то весь подалгоритм имеет время работы не более $\mathcal{O}^*(2^{\sigma|V(R)|})$ времени.

Если $R \setminus D$ содержит только лёгкие вершины, выдаём текущую пару (A, D) и прекращаем работу с данной подзадачей. Иначе граф $R \setminus D$ содержит тяжёлую вершину v. Вначале мы рассмотрим случай $v \notin A$. Связные компоненты $R[Y]$ содержат не более C вершин. Значит, если

$v \in Y$, то не более трети соседей вершины v из $R \setminus D$ принадлежат Y, поскольку $deg_{R\setminus D}(v) \geq 3C$. Следовательно, мы можем расщепиться ниже описанным образом. В первом случае добавим вершину v к множеству D. В другом случае добавим v к множеству A и переберём (создадим новые расщепления) все возможные разбиения (A', D') множества вершин $N_{R\setminus D}(v) \setminus A$ такие, что степень v в $A \cup A'$ не превосходит C. Для каждого из разбиений создадим новую подзадачу, в которой добавим вершины A' к множеству A и вершины D' к множеству D. Поскольку соседние вершины v в множестве $R \setminus D$ могут лежать в A или не принадлежать ни A, ни D, то не более трети вершин из $N_{R\setminus D}(v) \setminus A$ могут быть добавлены к множеству A. Отметим, что если $N_{R\setminus D}(v) \setminus A$ пусто, то вершина v уже содержит более C соседей, что противоречит предположению о размере компоненты и можно не рассматривать данную подзадачу. Отметим, что подобное расщепление корректно (существует некоторая пара (A, D) такая, что $A \subset Y$ и $Y \cap D = \emptyset$), поскольку все связные компоненты графа Y содержат не более C вершин.

Из леммы 2.2 следует комбинаторная оценка, которая будет полезна при доказательстве свойств потенциальной функции:

Факт 1. Если $|M| = m$, тогда число подмножеств множества M с мощностью не более $m/3$ не превосходит $2^{\sigma' \cdot m}$, где $\sigma' < 1$ — некоторая константа.

Действительно, мы можем взять σ' такое, что $2^{\sigma'} = 1.89$. Чтобы доказать, что суммарный потенциал все конечных подзадач не больше потенциала первоначальной задачи, достаточно проверить, что для $n = |V(R) \setminus (A \cup D)|$ и $m = |N_{R\setminus D}(v) \setminus A|$ верно следующее:

$$2^{\sigma n} \geq 2^{\sigma(n-1)} + 2^{\sigma' \cdot m} \cdot 2^{\sigma(n-1-m)}.$$

Это эквивалентно следующему неравенству:

$$2^{\sigma} \geq 1 + 2^{m(\sigma' - \sigma)}.$$

Мы выберем $1 > \sigma > \sigma'$ так, что $2^{\sigma}(2^{\sigma} - 1) \geq 2^{\sigma'}$. Заметим, что такие σ, σ' существуют, поскольку функция $f(t) = 2^t(2^t - 1)$ непрерывна и строго возрастает в окрестности 1 и $f(1) = 2 > 2^{\sigma'}$. Тогда

$$2^{\sigma} \geq 1 + 2^{\sigma' - \sigma} \geq 1 + 2^{m(\sigma' - \sigma)},$$

ведь $m \geq 1$ и $\sigma' < \sigma$. $\square$

В оставшемся случае, когда $v \in A$, мы просто не рассматриваем подслучай, в котором v добавлено к множеству D. Следовательно, чтобы доказать неравенство для потенциальной функции в этом случае достаточно проверить:

$$2^{\sigma n} \geq 2^{\sigma' \cdot m} \cdot 2^{\sigma \cdot (n-m)},$$

что верно так как $\sigma' < \sigma$. Тем самым мы доказали утверждение 2.2. $\square$

Вернёмся к доказательству леммы 2.5. Пусть

$$\mathcal{L} = \big\{(A_1, D_1), (A_2, D_2), \ldots, (A_p, D_p)\big\}$$

составляет множество пар, сгенерированных алгоритмом из утверждения 2.2. Мы знаем, что (i) $\sum_{i=1}^{p} \phi(A_i, D_i) \leq 2^{\sigma |V(R)|}$, (ii) для любого i все вершины в $R \setminus D_i$ лёгкие, (iii) существует индекс i_0 такой, что $A_{i_0} \subseteq Y$ и $D_{i_0} \cap Y = \emptyset$. Пусть $\mathcal{L}_{\mathrm{small}} \subset \mathcal{L}$ — множество, включающее только такие пары (A_i, D_i), что $|A_i| + |D_i| \geq \frac{|V(R)|}{4}$, а $\mathcal{L}_{\mathrm{large}} = \mathcal{L} \setminus \mathcal{L}_{\mathrm{small}}$. Сейчас для каждой пары $(A, D) \in \mathcal{L}$ мы выдадим список кандидатов на роль Y. Для пар из $\mathcal{L}_{\mathrm{small}}$ и $\mathcal{L}_{\mathrm{large}}$ мы применим разные подходы.

Для каждой пары из $(A, D) \in \mathcal{L}_{\mathrm{small}}$ переберём все возможные варианты. В качестве конечных кандидатов на роль Y, выдадим все множества вида $A \cup Y'$, где Y' — подмножество $V(R) \setminus (A \cup D)$. Очевидно, что если $(A_{i_0}, D_{i_0}) \in \mathcal{L}_{\mathrm{small}}$, то Y находится среди выданных кандидатов. Оценим количество выданных подмножеств кандидатов на роль Y в данном случае.

Для $(A, D) \in \mathcal{L}_{\mathrm{small}}$ положим $m = |V(R) \setminus (A \cup D)|$. Таким образом, для (A, D) выдадим ровно 2^m кандидатов. Поскольку $m \leq \frac{3}{4}|V(R)|$, получаем $2^m = 2^{\sigma m} \cdot 2^{(1-\sigma)m} \leq \phi(A, D) \cdot 2^{\frac{3}{4}|V(R)|(1-\sigma)}$. Тем самым, общее число кандидатов для пар из $\mathcal{L}_{\text{small}}$ не превосходит

$$\sum_{(A,D)\in\mathcal{L}_{\text{small}}} \phi(A,D)\cdot 2^{\frac{3}{4}|V(R)|(1-\sigma)} = 2^{\sigma|V(R)|+\frac{3}{4}|V(R)|(1-\sigma)} = 2^{\frac{3+\sigma}{4}|V(R)|}.$$

Заметим, что $\frac{3+\sigma}{4} < 1$ для $\sigma < 1$.

Сейчас осталось рассмотреть только пары из $\mathcal{L}_{\text{large}}$. Пусть $(A,D) \in \mathcal{L}_{\text{large}}$ и $Q = V(R) \setminus (A \cup D)$. Следующее утверждение играет ключевую роль для оставшейся части алгоритма и доказательства его корректности:

Лемма 2.6. Если $(A,D) \in \mathcal{L}_{\text{large}}$ и $A \subseteq Y$, $D \cap Y = \emptyset$, тогда $|Q \cap Y| \geq \frac{2}{3}|Q|$.

Другими словами, мы можем считать, что в правильной ветке расщепления как минимум $\frac{2}{3}$ всех оставшихся вершин принадлежат искомому решению. Прежде чем перейти к доказательству утверждения 2.6, покажем как использовать это утверждение, чтобы закончить алгоритм из леммы 2.5.

Для каждой пары $(A,D) \in \mathcal{L}_{\text{large}}$ переберём все возможные оставшиеся варианты распределения вершин между подмножествами, но при этом мы также воспользуемся утверждением 2.6. То есть, в качестве кандидатов мы выдадим все подмножества вида $A_i \cup Y'$, где $Y' \subset Q$ и $|Y'| \geq \frac{2}{3}|Q|$. Применяя факт 1 к дополнению Y', мы получаем, что общее число кандидатов не более $2^{\sigma' m} \leq 2^{\sigma m} = \phi(A,D)$, где $m = |Q|$. Таким образом, общее число кандидатов не превосходит:

$$\sum_{(A,D)\in\mathcal{L}_{\text{large}}} \phi(A,D) \leq 2^{\sigma|V(R)|} \leq 2^{\frac{3+\sigma}{4}|V(R)|}.$$

Таким образом, алгоритм выдаст не более $2\cdot 2^{\frac{3+\sigma}{4}|V(R)|}$ претендентов на роль Y: $2^{\frac{3+\sigma}{4}|V(R)|}$ для $\mathcal{L}_{\text{small}}$ и $2^{\frac{3+\sigma}{4}|V(R)|}$ для $\mathcal{L}_{\text{large}}$. Поэтому мы можем положить $\rho = \frac{3+\sigma}{4}$. Утверждение 2.6 гарантирует, что Y будет перечислено среди кандидатов для $\mathcal{L}_{\text{large}}$, если $(A_{i_0}, D_{i_0}) \in \mathcal{L}_{\text{large}}$. До этого мы доказали аналогичное утверждение, но при условии, что $(A_{i_0}, D_{i_0}) \in \mathcal{L}_{\text{small}}$.

Приступим к доказательству леммы 2.6.

Доказательство леммы 2.6. Предположим противное, то есть $|Q \cap Y| < \frac{2}{3}|Q|$. Поскольку $|Q| \geq \frac{3}{4}|V(R)|$, то получается:

$$|Q \setminus Y| > \frac{1}{3}|Q| \geq \frac{1}{4}|V(R)| \geq \frac{\gamma n}{8}.$$

Построим множество $T \subseteq Q \setminus Y$ со следующими свойствами:

- T — независимое множество в G,
- никакие две вершины из T не соединены с одной и той же компонентой связности из $R[Y]$,
- $|T| \geq \frac{\gamma n}{104C^3}$.

Множество T построим, используя жадную стратегию. Мы последовательно добавляем к T неиспользованную вершину v из $Q \setminus Y$, после чего следующие вершины из $Q \setminus Y$ помечаем как использованные: (i) саму вершину v, (ii) всех соседей вершины v в графе $Q \setminus Y$ и (iii) все вершины из множества $Q \setminus Y$, которые смежны хотя бы с одной связной компонентой графа $R[Y]$, соединённой с v. Напомним, что степени вершин в графе $R \setminus D$ ограничены $3C$ и $Q \subseteq V(R) \setminus D$, таким образом, v смежна не более чем с $3C$ вершинами в $Q \setminus Y$. Аналогично, v соединено не более чем с $3C$ связными компонентами из $R[Y]$. Размер каждой компоненты не превосходит C, более того, каждая вершина из любой компоненты смежна не более чем с $3C$ вершинами из $Q \setminus Y$. Всего за один раз мы пометили как использованные не более $1 + 3C + 3C \cdot C \cdot 3C \leq 13C^3$ вершин. Следовательно, мы сможем находить неиспользованную вершину на протяжении как минимум $\frac{|Q \setminus Y|}{13C^3} \geq \frac{\gamma n}{104C^3}$ раундов. Легко видеть, что все свойства для T выполняются по построению.

Утверждается, что $R[T \cup Y] \in \Pi$. Поскольку для любой вершины $v \in T$ $deg_{R \setminus D}(v) \leq 3C$, то получается, что мощности связных компонент $R[T \cup Y]$ не превосходят $1 + 3C^2$. Если $R[T \cup Y] \notin \Pi$, тогда какой-то запрещённый граф из множества $\mathcal{F}_\Pi$ содержится в $R[T \cup Y]$. Поскольку все графы в $\mathcal{F}_\Pi$ связны, такой запрещённый подграф должен содержаться в одной связной компоненте $R[T \cup Y]$, а значит, и состоять не более чем из $1 + 3C^2$ вершин. Но это противоречит нашему предположению, что G не содержит индуцированных подграфов из $\mathcal{F}_\Pi$ такого размера. Следовательно, $R[T \cup Y] \in \Pi$.

Осталось показать, что подобная конструкция подграфа T противоречит тому, что H — это максимальный индуцированный Π-подграф. Рассмотрим индуцированный подграф $H' = G[X \cup Y \cup T]$. Другими словами, удалим клику S из решения H и добавим множество вершин T. Очевидно, H' — это объединение непересекающихся графов $G[X]$ и $G[Y \cup T]$. Поскольку оба графа принадлежат классу Π, то и их объединение H' тоже лежит в Π. Более того, поскольку $|S| = \alpha n < \frac{\gamma n}{104C^3} \leq |T|$, получаем, что $|V(H')| > |V(H)|$. Тем самым мы получили противоречие с оптимальностью H.

$\square$

Как было замечено ранее, утверждение 2.6 завершает доказательство леммы 2.5. $\square$

2.4 Описание порядка выбора констант

В этом разделе мы представим краткое описание того, как выбрать значения констант. Назначение констант описано в таблице 2.1. В последующем под *быстрее* 2^n мы имеем в виду время работы $\mathcal{O}^*(2^{\kappa n})$, где $\kappa < 1$.

Сперва обсудим случай B.1.3. В данном случае время работы алгоритма составляет $\mathcal{O}^*(\binom{n}{\varepsilon n} \cdot \binom{n}{\zeta n}) \cdot \mathcal{O}^*(2^{\kappa_7 n})$ для некоторой константы $\kappa_7 < 1$ такой, что $\binom{n}{\frac{3}{8}n} = \mathcal{O}^*(2^{\kappa_7 n})$. Следовательно, мы можем найти верхнюю границу $\epsilon_0 > 0$ на значения $\alpha, \beta, \delta, \varepsilon$ так, что взяв любое значение этих констант меньше ϵ_0, получим время работы в случае B.1.3 быстрее чем 2^n.

Случай B.1.2. Вначале зафиксируем значение $\varepsilon > 0$ так, что $\varepsilon < \epsilon_0$. Как было отмечено ранее, для фиксированного $\varepsilon > 0$ можно найти положительную верхнюю границу $\epsilon_1 < \epsilon_0$ на значения $\alpha, \beta, \gamma, \delta$ такую,

Таблица 2.1: Краткое описание используемых констант.

Константа	Краткое описание	Определено в
α	наибольший размер клики в графе G	Шаг 1
β	размер решения лежит в интервале $[\frac{n}{2} - \beta n, \frac{n}{2} + \beta n]$	Шаг 2
γ	число связных компонент в $H \setminus S$	Шаг 4
δ	$\big\|\|N'(X_1)\| - \|X_1\|\big\| \geq \delta n$	Случай B.1.1
ε	$\|N'(X_1) \cap N'(X_2)\| \geq \varepsilon n$	Случай B.1.2
ζ	$2\alpha + 2\beta + 2\delta + \varepsilon$, $\|U_{\text{none}}\| \leq \zeta$	Случай B.1.3
L	большая константа > 2	Случай B.2
C	$C = \frac{L}{\gamma}$, размер маленьких компонент в $H \setminus S$	Случай B.2
ℓ	граф G поданный на вход не содержит запрещённых индуцированных подграфов из множества $\mathcal{F}_\Pi$, чей размер не превосходит ℓ	Перед шагом 1
ℓ	равно $3C^2 + 1$	Конец алгоритма

что любые значение $\alpha, \beta, \gamma, \delta$ меньшее ϵ_1 приведёт к работе алгоритма быстрее 2^n в этом случае.

Аналогично рассмотрим случай B.1.1. Зафиксируем произвольное $0 < \delta < \epsilon_1$. Ранее было показано, что для фиксированного $\delta > 0$ можно найти положительную верхнюю границу $\epsilon_2 < \epsilon_1$ так, что при значениях α, γ меньше ϵ_2, алгоритм отработает в этом случае быстрее 2^n.

Рассмотрим ситуацию в ветви B.2. Выберем любое $\gamma > 0$ такое, что $\gamma < \epsilon_2$. Напоним, что время работы в этой ветви составило $\mathcal{O}^*(\binom{n}{\alpha n} \cdot \binom{n}{\frac{\gamma n}{L}} \cdot 2^{|N'[X]|} \cdot 2^{\rho|V(R)|})$, где $|N'[X]| + |V(R)| \leq n$, $|V(R)| \geq \frac{\gamma n}{2}$ и $\rho < 1$ — некоторая универсальная константа. Поэтому при заданном $\gamma > 0$ можно найти нижнюю границу $L_0 \geq 2$ на значение L и верхнюю границу $\epsilon_3 < \epsilon_2$ на значение α такие, что для любых $L > L_0$ и положительного $\alpha < \epsilon_3$ получим время работы быстрее 2^n. Мы фиксируем произвольное $L > L_0$ и уменьшаем значение ϵ_3, если необходимо для справедливости неравенства $\epsilon_3 < \frac{\gamma}{104C^3}$, где $C = L/\gamma$. После этого можно выбрать значение двух оставшихся констант: возьмём β меньше ϵ_1

так, чтобы шаг 2 требовал времени меньше чем 2^n и α положим равным любому числу меньше ϵ_3. Таким образом в обоих ветвях B.1 и B.2 мы получили время работы быстрее 2^n.

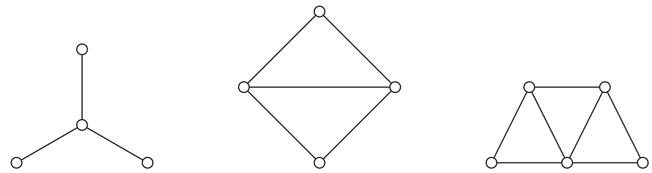
Поскольку случай A работает быстрее 2^n для любого $\alpha > 0$, точнее за время $\mathcal{O}^*(2^{(1-(1-\kappa_0)\alpha)n})$ для некоторого $\kappa_0 < 1$, зависящего от $\aleph$, мы получаем, что весь алгоритм работает быстрее 2^n. Но мы предположили, что алгоритм получает на вход только $\mathcal{F}'_\Pi$-свободные графы, где $\mathcal{F}'_\Pi$ состоит из графов $\mathcal{F}_\Pi$ размера не больше ℓ, для $\ell = 3C^2 + 1 = 3\frac{L^2}{\gamma^2} + 1$. Поскольку $\mathcal{F}'_\Pi$ — это конечное множество графов, можно применить лемму 2.1 считая, что класс Π — это класс всех графов.

Отметим, что ℓ квадратично зависит от C, которое в свою очередь зависит от L и γ. Поскольку L зависит от γ, а γ в свою очередь зависит от δ и ε, то даже очень грубые подсчёты показывают, что значение ℓ будет порядка нескольких тысяч или даже больше. Таким образом уже в самом начале алгоритма мы расщепляемся по запрещённым графам, чей размер достигает нескольких тысяч. При расщеплении удаляется всего один подслучай из 2^ℓ, в котором все вершины принадлежат подграфу H. Таким образом, если бы мы решили представить κ как $0.9999\ldots99$, тогда число девяток было бы не меньше чем два в степени нескольких тысяч. По этой причине вычисление точных значений констант не производилось.

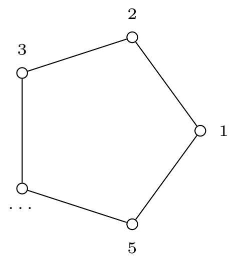
2.5 Заключение

Теорема 2.1 показывает, что для любого класса графов, удовлетворяющего свойствам (1)–(4), максимальный индуцированный Π-подграф в графе из n вершин может быть найден за время $\mathcal{O}^*(2^{\lambda n})$ для некоторого $\lambda < 1$. Объединив результаты леммы 2.1 и теоремы 2.1, получаем, что если кроме принадлежности классу Π мы запретим некоторое конечное число индуцированных подграфов, то такой подграф все равно может быть найден быстрее полного перебора. Таким образом, верна следующая теорема:

Теорема 2.3. Пусть $\mathcal{F}$ — конечное множество графов и Π — класс графов, удовлетворяющий свойствам (1)–(4). Существует алгоритм, который в заданном графе G на n вершинах находит $\mathcal{F}$-свободный индуцированный Π-подграф за время $\mathcal{O}^*(2^{\lambda n})$ для некоторого $\lambda < 1$, где λ зависит только от $\aleph$ и $\mathcal{F}$.



(a) Клешня (b) Алмаз (c) Самоцвет



(d) Дыра (цикл)

Рис. 2.5: Запрещенные графы для некоторых классов графов.

Отметим, что теорема 2.3 включает в себя такие классы графов, как собственные интервальные графы (интервальные графы без клешней, см. рис. 2.5), графы Птолемея (хордальные графы без самоцветов, см. рис. 2.5), блоковые графы (хордальные графы без алмазов, см. рис. 2.5).

Результаты данной главы опубликованы в работах [27, 44].

Глава 3

Нижние оценки на вычислительную сложность задачи о рёберных дополнениях

3.1 Условная нижняя оценка на вычислительную сложность задачи оптимального линейного упорядочивания

В данном разделе докажем нижнюю оценку для задачи об оптимальном линейном упорядочивании при условии справедливости гипотезы ETH.

Прежде всего напомним формулировку задачи об оптимальном линейном упорядочивании и приведем необходимые сведения о графах-экспандерах.

Число Чигера $h(G)$ графа G определяется следующим образом:

$$h(G) := \min\left\{\frac{|\delta(X)|}{|X|} : X \subseteq V(G), |X| \leq \frac{|V(G)|}{2}\right\},$$

Граф G является (d, e)-*экспандером*, если он является d-регулярным и его число Чигера не меньше e. При работе с экспандерами мы допускаем присутствие кратных рёбер и петель, поскольку подобные ребра естественным образом присутствуют в конструкциях экспандеров. Будем считать, что петля увеличивает степень вершины на единицу.

При построении некоторых сведений в работе будет использована следующая теорема.

Теорема 3.1. [Теорема 21.19 из [54]] Пусть $p > 0$ — действительное число. Тогда существует положительное целое d, такое что для любого положительного целого n существует d-регулярный мультиграф $G_{n,d}$ на n вершинах и $h(G_{n,d}) \geq p$. Более того, граф $G_{n,d}$ может быть построен за полиномиальное время от n.

Под линейным упорядочиванием графа G мы понимаем функцию $\pi : V(G) \to \{1, 2, \ldots n\}$, стоимостью упорядочивания называется величина $\sum_{uv \in E} |\pi(u) - \pi(v)|$. Будем говорить, что упорядочивание π графа G оптимально, если не существует упорядочивания с меньшей стоимостью. Стоимость упорядочивания π обозначим через ОЛУ(π), стоимость оптимального упорядочивания графа G — через ОЛУ(G). В задаче об оптимальном линейном упорядочивании задан граф $G = (V, E)$ и целое число k, требуется определить, существует ли линейное упорядочивание графа G со стоимостью не более k.

Главным результатом данного раздела является доказательство следующей теоремы.

Теорема 3.2. Если гипотеза ETH верна, то существует такое $c > 1$, что задача оптимального линейного упорядочивания не допускает алгоритма с временем работы $2^{\mathcal{O}(n/\log^c n)}$.

Отметим, что задача оптимального линейного упорядочивания может быть решена с помощью метода динамического программирования по подмножествам за время $2^n \cdot n^{\mathcal{O}(1)}$, таким образом теорема 3.2 не может быть значительно улучшена.

В работе [45] доказана следующая теорема.

Теорема 3.3. Если справедлива гипотеза ETH, то существуют такие $c \in \mathbb{N}$ и $r \in (0, 1)$, что задача Заз-Р3-ВЫП$_{[r,1]}$ не допускает алгоритма с временем работы $2^{\mathcal{O}(\frac{m}{\log^c(m)})}$.

Сведём задачу Заз-Р3-ВЫП$_{[\alpha,\beta]}$ к задаче о максимальном разрезе с зазором. Для этого мы используем три стандартных сведения, первоначально использовавшиеся для доказательства NP трудности в работе [55]. Каждое из сведений выдаёт экземпляр лишь в константу раз большего размера первоначальной задачи и, более того, все сведения сохраняют зазор.

Теорема 3.4.

$$\text{Заз-Р3-ВЫП}_{[\alpha,\beta]} \leq_{\mathrm{P}}^{\mathrm{lin}} \text{Заз-Р4-НВ-ВЫП}_{[\alpha,\beta]} \leq_{\mathrm{P}}^{\mathrm{lin}}$$
$$\text{Заз-Р3-НВ-ВЫП}_{[\frac{1+\alpha}{2},\frac{1+\beta}{2}]} \leq_{\mathrm{P}}^{\mathrm{lin}}$$
$$\text{максимальный разрез с зазором}_{[\frac{16+\alpha}{18},\frac{16+\beta}{18}]}$$

Разобьём доказательство теоремы на четыре леммы.

Лемма 3.1.

$$\text{Заз-Р3-ВЫП}_{[\alpha,\beta]} \leq_{\mathrm{P}}^{\mathrm{lin}} \text{Заз-Р4-НВ-ВЫП}_{[\alpha,\beta]}$$

Доказательство. По заданной Р3-КНФ формуле $\phi = C_1 \wedge \ldots \wedge C_m$ построим Р4-КНФ формулу $\phi' = C_1' \wedge \ldots \wedge C_m'$. Добавим новую переменную z к каждому клозу. То есть для клоза $C_i = l_1 \vee l_2 \vee l_3$, построим клоз $C_i' = l_1 \vee l_2 \vee l_3 \vee z$.

Если какое-то означивание выполняет k клозов в формуле ϕ, то положив $z = 0$ мы получим, что k соответствующих клозов симметрично выполнено в формуле ϕ'. С другой стороны, если означивание φ симметрично выполняет k клозов формулы ϕ', то его отрицание $\tilde{\varphi}$ также симметрично выполняет k клозов ϕ'. Поэтому, не ограничив общности, можно считать, что в означивании φ переменная z принимает значение 0. А значит, означивание φ, ограниченное на переменные формулы ϕ, выполняет k клозов в формуле ϕ. Легко видеть, что при таком сведении зазор сохраняется. $\square$

Лемма 3.2.

$$\text{Заз-Р4-НВ-ВЫП}_{[\alpha,\beta]} \leq_{\mathrm{P}}^{\mathrm{lin}} \text{Заз-Р3-НВ-ВЫП}_{[\frac{1+\alpha}{2},\frac{1+\beta}{2}]}$$

Доказательство. К заданной Р4-КНФ формуле $\phi = C_1 \wedge ... \wedge C_m$ добавим m новых переменных $z_1, ..., z_m$ и заменим каждый клоз $C_i = l_1 \vee l_2 \vee l_3 \vee l_4$ на два новых: $C_i' = l_1 \vee l_2 \vee z_i$, $C_i'' = l_3 \vee l_4 \vee \neg z_i$. Проделав такую операцию, получим Р3-КНФ формулу ϕ'. Покажем, что можно симметрично выполнить не менее k клозов в ϕ тогда и только тогда, когда можно симметрично выполнить не менее $m+k$ клозов в формуле ϕ'. Отметим, что этого достаточно для доказательства теоремы.

Заметим, что если означивание симметрично выполняет клоз C_i, то можно выбрать значение переменной z_i так, чтобы оба клоза C_i' и C_i'' были симметрично выполнены. С другой стороны, если при означивании клоз C_i не выполнен симметрично, то при любом значении z_i ровно один клоз из C_i' и C_i'' будет симметрично выполнен. Таким образом, любое означивание, которое симметрично выполняет k клозов ϕ, можно продолжить до означивания переменных формулы ϕ' так, чтобы было симметрично выполнено $m+k$ клозов в формуле ϕ'.

Докажем в обратную сторону. Предположим, что существует означивающий набор φ, который симметрично выполняет не меньше $m+k$ клозов ϕ'. Заметим, что существует такое множество I, состоящее как минимум из k индексов i, что φ симметрично выполняет оба клоза C_i' и C_i''. Рассмотрим фиксированный индекс $i \in I$, предположим в означивании φ переменная z_i принимает значение 0 (случай со значением 1 рассматривается аналогично). Означивание φ симметрично выполняет оба клоза C_i' и C_i''. А значит один из литералов l_1, l_2 в φ принимает значение 1 и как минимум один из литералов l_3, l_4 принимает значение 0. Таким образом, φ ограниченное только на переменные формулы ϕ, симметрично выполняет все клозы C_i при $i \in I$, что завершает доказательство теоремы, поскольку $|I| \geq k$. □

Лемма 3.3. Существует полиномиальное по времени линейное сведение от задачи Заз-Р3-НВ-ВЫП$_{[\alpha,\beta]}$ к задаче о максимальном разрезе на мультиграфах. При этом зазор $[\alpha, \beta]$ переходит в зазор $[\frac{3+2\alpha}{6}, \frac{3+2\beta}{6}]$.

Доказательство. Рассмотрим заданную Р3-КНФ формулу $\phi = C_1 \wedge$

$\ldots \wedge C_m$. Пусть n_i обозначает количество вхождений переменной x_i в ϕ, учитывая положительные и отрицательные литералы. Построим мультиграф G с $2n$ вершинами и $6m$ рёбрами следующим образом. Для каждой переменной x_i создадим две вершины, соответствующие литералам x_i и $\neg x_i$. После этого добавим n_i рёбер между ними и для каждого клоза C_i построим треугольник, соединяющий вершины, соответствующие литералам из клоза C_i. Покажем, что существует означивание ϕ симметрично выполняющее не менее k клозов тогда и только тогда, когда в G существует разрез, содержащий не менее $3m + 2k$ рёбер.

Предположим, что существует означивание, которое симметрично выполняет k клозов в формуле ϕ. Построим разрез в графе G следующим образом: с одной стороны расположим вершины, соответствующие литералам, принимающим значение 1, а с другой стороны — все остальные вершины. Легко видеть, что любое ребро, соединяющее x_i и $\neg x_i$, находится в разрезе. Суммарно у нас всего $\sum_i n_i = 3m$ таких рёбер. Более того, из каждого треугольника, соответствующего симметрично выполненному клозу, в разрезе участвует ровно два ребра. Поскольку выполнено не менее k клозов, получается, что размер разреза не менее $3m+2k$. Докажем утверждение в обратную сторону. Предположим, в G есть разрез размера $3m + 2k$ или больше. Допустим, существует такая переменная x_i, что соответствующие ей вершины $v_{x_i}, v_{\neg x_i}$ находятся на одной стороне разреза. Заметим, что $N(v_{x_i}, v_{\neg x_i}) \leq n_i$, а значит, переместив v_{x_i} или $v_{\neg x_i}$ в другую сторону, мы не уменьшим разрез. Таким образом, не ограничивая общности, мы можем предположить, что для любого j вершины v_{x_j} и $v_{\neg x_j}$ находятся по разные стороны разреза. Следовательно, вклад в размер разреза рёбер, соединяющих v_{x_j} и $v_{\neg x_j}$, равен $3m$. Если в треугольнике, соответствующему клозу, два ребра участвуют в разрезе, то такой клоз симметрично выполнен, иначе клоз не выполнен симметрично, поскольку все его литералы имеют одно и то же значение. Учитывая этот факт, по заданному разрезу размера $3m + 2k$ легко построить означивание, симметрично выполняющее не менее k клозов в формуле ϕ. $\square$

Лемма 3.4. Существует полиномиальное по времени линейное сведение от задачи о максимальном разрезе на мультиграфах с зазором к задаче о максимальном разрезе с зазором, при этом зазор $[\alpha, \beta]$ переходит в зазор $[\frac{2+\alpha}{3}, \frac{2+\beta}{3}]$.

Доказательство. Преобразуем задачу максимального разреза на мультиграфах в задачу о максимальном разрезе на графах без кратных рёбер и петель. По графу $G = (V, E)$ построим граф $G' = (V', E')$, где $V' = V \cup \{w_e, z_e | e \in E\}$ и $E' = \{uw_e, w_e z_e, z_e v | e = uv, e \in E\}$. Другими словами, для каждого ребра $e = uv$ мультиграфа G создаём две вспомогательные вершины w_e, z_e и три ребра uw_e, $w_e z_e$, $z_e v$ (отметим, что ребро uv не добавлено в граф G'). Мы утверждаем, что граф G содержит разрез размера k тогда и только тогда, когда граф G' содержит разрез размера $2m + k$.

Пусть разрез C графа G содержит k рёбер. Если $e = uv \notin C$, мы поместим вершины w_e и z_e на сторону, противоположную вершине u. В таком случае два ребра uw_e и $z_e v$ будут участвовать в разрезе графа G'. Если $e = uv \in C$, то разместив w_e, z_e в противоположные стороны от u, v соответственно, получим три новых ребра в разрезе графа G'. Суммируя, получаем, что размер разреза равняется $2m + k$.

Пусть C — разрез графа G' и $|C| = 2m + k$, тогда существует как минимум k рёбер uv из первоначального графа G таких, что все три ребра uw_e, $w_e z_e$, zv_e находятся в разрезе C. Заметим, что для такого ребра uv вершина w_e находится на отличной стороне от вершины u, z_e на отличной стороне от w_e и вершины v, z_e тоже находятся на разных сторонах. Получается, что v и u находятся на разных сторонах. Следовательно, ограничение разреза только на вершины графа G даёт нам разрез размера не менее k в графе G. □

Из лемм 3.1, 3.2, 3.3, и 3.4 вытекает справедливость теоремы 3.4. Отметим, что из теоремы 3.4 немедленно следует теорема:

Теорема 3.5. Если гипотеза ETH верна, то найдутся такие $c \in \mathbb{N}$ и $0 \leq \alpha < \beta \leq 1$, что не существует $2^{\mathcal{O}(\frac{m}{\log^c(m)})}$-алгоритма для задачи о максимальном разрезе с зазором$_{[\alpha,\beta]}$.

Приведём сведение от задачи о максимальном разрезе с зазором к задаче об оптимальном линейном упорядочивании. Наше сведение является модификацией сведения, представленного Гэри, Джонсоном и Стокмейером в работе [56]. Их сведение преобразовывает задачу о максимальном разрезе в задачу об оптимальном линейном упорядочивании. В конструкции [56] добавляется клика большого размера к графу $\overline{G}$ (дополнение графа G), где граф G — это граф из экземпляра задачи максимальный разрез. Цель клики заключается в разделении сторон максимального разреза в оптимальном линейном упорядочивании. Другими словами, мы хотим, чтобы в оптимальном упорядочивании первоначально шли вершины одной стороны, затем вершины из клики и только после этого вершины другой стороны разреза. Размер клики выбирается большой, чтобы вклад любого ребра из разреза превосходил "помехи", создаваемые внутренним упорядочиванием сторон. Для выполнения этой цели размер клики в [56] выбран равным $\Theta(n^4)$, что влечёт к значительному увеличению размера выдаваемого экземпляра. А значит, не позволяет доказать нижнюю субэкспоненциальную оценку на время вычисления для задачи об оптимальном линейном упорядочивании.

Именно по этой причине мы используем в сведении вместо задачи о максимальном разрезе задачу о максимальном разрезе с зазором. В некотором смысле с помощью зазора мы боремся с "помехами" внутренних рёбер, что позволяет использовать нам клику лишь линейного размера. Отметим, что хоть в построенном сведении мы получим линейное число вершин, количество рёбер будет квадратичным.

Теорема 3.6. Существует полиномиальный алгоритм, который по заданному экземпляру I задачи о максимальном разрезе с зазором$_{[\alpha,\beta]}$ на n вершинах выдаёт экземпляр I' задачи об оптимальном линейном упорядочивании с $\mathcal{O}(\frac{n}{\beta-\alpha})$ вершинами. При этом, если I содержит разрез с не менее βm рёбрами, то стоимость линейного упорядочивания I' не превосходит некоторого значения, если I не допускает разрезов с числом рёбер больше αm, то стоимость I' превосходит заданное значение.

Доказательство. Для заданного на входе графа G, мы создадим соответствующий экземпляр G' задачи об оптимальном линейном упорядочивании следующим образом. Рассмотрим дополнение графа G, то есть граф $\overline{G}$, и добавим к нему клику C размера Mn, где $M = \lceil \frac{2}{\beta - \alpha} \rceil$. Полностью соединим клику с графом $\overline{G}$, то есть каждую вершину клики соединим с каждой вершиной $\overline{G}$).

Покажем, что G допускает разрез размера не меньше βm тогда и только тогда, когда существует линейное упорядочивание G' стоимостью не более $\binom{(M+1)n+1}{3} - \beta m \cdot Mn$, тем самым мы докажем эквивалентность экземпляров.

Пусть (A, B) — разрез размера не меньше βm. Зададим линейное упорядочивание π следующим образом. Вначале перечислим все вершины множества A, затем мы перечислим вершины клики C, а после этого вершины множества B. Порядок вершин внутри множеств A, B и C может быть произвольным. Вычисляя стоимость всех ребер в полном графе мы получим равенство:

$$\sum_{uv \in E(G')} |\pi(u) - \pi(v)| + \sum_{uv \in E(G)} |\pi(u) - \pi(v)| = \binom{(M+1)n+1}{3} \tag{3.1}$$

Стоимость любого ребра, проходящего над кликой (ребра из $E_{G'}(A, B)$), не меньше Mn, а значит $\sum_{uv \in E(G)} |\pi(u) - \pi(v)| \geq \beta m \cdot Mn$, что влечёт требуемое неравенство.

Докажем утверждение в обратную сторону. Пусть π — это линейное упорядочивание графа G' стоимостью не более $\binom{(M+1)n+1}{3} - \beta m \cdot Mn$. Для начала докажем, что существует такое оптимальное линейное упорядочивание π, в котором все вершины клики C расположены последовательно, то есть цельным блоком в π. Прежде чем приступить к доказательству, введём несколько определений, связанных с линейным упорядочиванием.

Будем отождествлять упорядочивание π с последовательностью $\pi(1), \pi(2), \ldots, \pi(n)$. Вершину, отображённую перестановкой π в i, будем называть i-ой вершиной (слева). Будем говорить, что множество вершин U расположено *последовательно в* π, если $\pi(U) = \{p, p{+}1, \ldots q{-}1, q\}$ для некоторых $p, q \in \mathcal{N}$. Множество всех вершин, расположенных в перестановке π перед множеством (слева от множества) U, будем обозначать $L(U)$. Аналогично обозначим множество всех вершин после (справа от) U через $R(U)$. *Блоком* множества U назовём произвольное последовательное максимальное по включению подмножество U. *Левым блоком* множества U назовём блок множества U, отображённый перестановкой π в наименьшие значения среди $\pi(U)$. *Вторым левым* блоком U назовём самый первый блок U, расположенный справа от левого блока множества U. Множество вершин из множества $V(G') \setminus U$, расположенных между левым и вторым левым блоком, назовём *внутренним блоком* множества U (если U состоит их одного блока, то внутренний блок не существует).

Утверждение 3.7. Существует оптимальное линейное упорядочивание π графа G', в котором вершины клики C расположены последовательно.

Доказательство. Рассмотрим оптимальное линейное упорядочивание π графа G' с минимальным числом вершин из $V(G)$, расположенных между вершинами из клики C. То есть такое упорядочивание, на котором функция $f(\pi) = |\{v \in V(G) : \exists u, w \in V(C), \pi(u) < \pi(v) < \pi(w)\}|$ достигает минимального значения. Утверждается, что в таком упорядочивание вершины клики C расположены последовательно, то есть $f(\pi) = 0$.

Предположим противное. Пусть X является внутренним блоком C. Рассмотрим два случая:

- $|E_G(L(X), X)| \leq |E_G(X, R(X))|$
- $|E_G(L(X), X)| > |E_G(X, R(X))|$

В каждом из случаев построим новое линейное упорядочивание, на котором функция f будет принимать значение меньшее $f(\pi)$, что противоречит предположению о минимальности π.

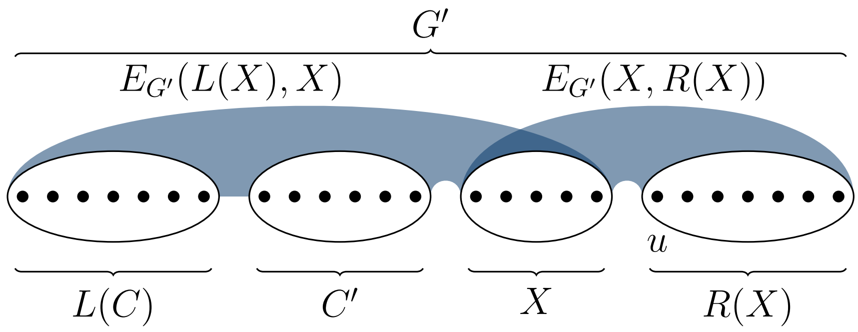


Рис. 3.1: Расположение вершин в доказательстве теоремы 3.7.

Рассмотрим первый случай, пусть C' — это левый блок C. Мы поменяем местами C' и X. Проанализируем, как изменится стоимость упорядочения π после такой операции. Заметим сперва, что из формулы (3.1) следует, что оптимальность π эквивалентна максимизации общей стоимости рёбер подграфа G, то есть π оптимально тогда, когда $\sum_{uv \in E(G)} |\pi(u) - \pi(v)|$ принимает наибольшее значение. Утверждается, что значение этой суммы не уменьшилось. Ниже в доказательстве под рёбрами мы будем иметь в виду только рёбра графа G (отсутствующие рёбра в G').

Очевидно, что стоимость рёбер, полностью лежащих в $V(G) \setminus X$ или X, не изменилась. А значит, достаточно исследовать поведение рёбер, содержащих ровно одну вершину в X. Любое такое ребро идёт из X в $L(X)$ или из X в $R(X)$. Длина ребра, идущего из X в $L(X)$, уменьшилась на $|C|$, а длина ребра, идущего из X в $R(X)$, увеличилась на $|C|$. Поскольку $|E_G(L(X), X)| \le |E_G(X, R(X))|$, то получается, что суммарная длина рассматриваемых рёбер не уменьшилась. При этом число вершин между вершинами клики C уменьшилось на $|X|$, что противоречит выбору упорядочивания.

Рассмотрим второй случай. Пусть C' — это второй левый блок C. Аналогично предыдущему случаю, мы поменяем местами блоки C' и X. Как и раньше длина изменится только у рёбер идущих из X в $G \setminus X$.

Из неравенства $|E_G(L(X),X)| > |E_G(X,R(X))|$ следует, что значение $\sum_{uv\in E(G)} |\pi(u)-\pi(v)|$ увеличилось на $|X|$, что противоречит оптимальности π.

Таким образом, мы доказали, что существует оптимальное линейное упорядочивание, в котором вершины C идут последовательно.

□

Пусть π — оптимальное линейное упорядочивание, в котором вершины клики C идут последовательно. Существование такого упорядочивания следует из только что доказанной леммы. Построим разрез (A, B). Возьмём в качестве A множество вершин $L(C)$, а в качестве B множество $R(C)$. Из (3.1) следует:

$$\beta m \cdot Mn \le \sum_{uv\in E(G)} |\pi(u)-\pi(v)| =$$

$$\sum_{\substack{uv\in E(G),\\ u,v\in A}} |\pi(u)-\pi(v)| + \sum_{\substack{uv\in E(G),\\ u,v\in B}} |\pi(u)-\pi(v)| + \sum_{\substack{uv\in E(G),\\ u\in A, v\in B}} |\pi(u)-\pi(v)| <$$

$$nm + (Mn+n)\cdot |E_G(A,B)| \le 2nm + Mn \cdot |E_G(A,B)|$$

А значит,

$$|E_G(A,B)| > \beta m - \frac{2m}{M} \ge \beta m - (\beta-\alpha)\cdot m = \alpha m$$

Нам известно, что если G содержит разрез размера больше αm, то в этом графе также есть разрез с количеством рёбер не меньше βm.

□

Доказательство теоремы 3.2 немедленно вытекает из теорем 3.5 и 3.6.

3.2 Разреженное сведение

В этом разделе мы представим полиномиальное сведение по Тьюрингу задачи о минимальной бисекции к задаче оптимального линейного упорядочивания на d-регулярных графах. Важным свойством данного сведения является то, что выданный граф имеет ограниченную степень и линейное число вершин по сравнению с первоначальным графом. Отметим, что выданный граф может содержать кратные ребра. Построенное сведение позволяет, посредством решения задачи об оптимальном линейном упорядочивании, различать экземпляры задачи о минимальной бисекции с разрезами, содержащими не более αm рёбер, и экземпляры с разрезами содержащими не менее βm рёбер, где $0 \leq \alpha < \beta \leq 1$ — некоторые заранее зафиксированные константы. Данный результат доказывает нижнюю оценку для задачи об оптимальном линейном упорядочивании в предположении следующей гипотезы.

Гипотеза 3.2. Существуют такие вещественные $0 \leq \alpha < \beta \leq 1$ и натуральное $d \geq \frac{4}{\beta - \alpha}$, что задача о минимальной бисекции с зазором$(d)_{[\alpha,\beta]}$ не допускает алгоритма с временем работы $2^{o(n)}$.

Главным результатом данного раздела является следующая теорема:

Теорема 3.8. Если гипотеза 3.2 верна, то существует такое $d \in \mathbb{N}$, что задача об оптимальном линейном упорядочивании ОЛУ$_{\leq}(d)$ на мультиграфах (со степенью не больше d) не допускает алгоритма с временем работы $2^{o(n)}$.

Вначале мы опишем преобразование $T(\cdot)$, которое переводит экземпляр задачи о минимальной бисекции в экземпляр задачи об оптимальном линейном упорядочивании. Данное преобразование составляет основу нашего сведения. Сперва мы опишем важные свойства преобразования $T(\cdot)$. А затем покажем, как классифицировать размер разреза в задаче о минимальной бисекции (меньше αm или больше βm), основываясь на стоимости оптимального линейного упорядочивания графа $T(G)$. Тем самым мы докажем теорему 3.8. Для доказательства нижней оценки $2^{\Omega(n)}$ важно, что трансформация $T(\cdot)$ лишь линейно увеличивает размер графа. Этим фактом также будем пользоваться при последующих сведениях.

Результат преобразования зависит от нескольких параметров. Точные значения параметров будут указаны позже, чтобы не препятствовать восприятию главных идей преобразования.

Рассмотрим экземпляр G задачи о минимальной бисекции, где G — это d_G-регулярный граф. Пусть $V(G) = \{v_1, v_2, \ldots, v_n\}$. Преобразование T выдаёт граф $G' = T(G)$ с множеством вершин $\{v_1, v_2, \ldots, v_n, x_1, x_2, \ldots x_{Z \cdot \lceil \varphi n \rceil}\}$, точные значения параметров $Z \in \mathbb{N}$ и $\varphi \in (0, 1)$ будут указаны позже. Отметим, что G' содержит вершины графа G. Более того, ребра в графе G' будут выбраны таким образом, чтобы граф G являлся индуцированным подграфом G'.

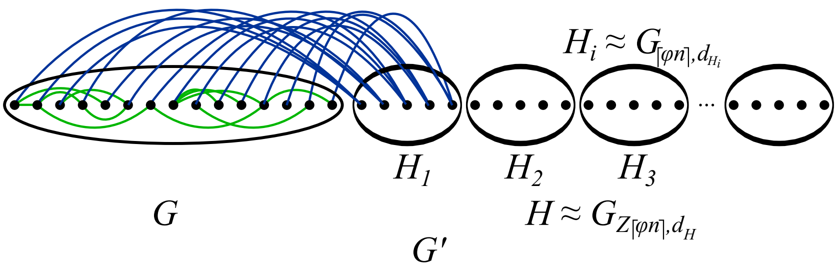


Рис. 3.2: Экземпляр задачи об оптимальном линейном упорядочивании $G' = T(G)$, полученный применением преобразования. Первоначальный граф G является индуцированным подграфом G', его ребра нарисованы зелёным. Ребра, добавленные между $V(G)$ и $V(H_1)$, окрашены синим цветом.

Для удобства введём обозначения для некоторых индуцированных подграфов G'. Индуцированный подграф на вершинах $x_1, x_2, \ldots, x_{z \cdot \lceil \phi n \rceil}$ обозначим через H. Подграф H произвольным образом разбит на Z непересекающихся индуцированных подграфов H_i размера $\lceil \varphi n \rceil$ каждый, для фиксированных параметров $Z \in \mathbf{N}$ и $\varphi \in (0, 1)$.

Результат преобразования представлен на рисунке 3.2. Рёберное множество графа G' построено следующим образом:

- Индуцированный подграф G' на вершинах $\{v_1, v_2, \ldots v_n\}$ порождает граф G.

- Используя теорему 3.1, мы построим d_H-регулярный экспандер

$G_{|H|,d_H}$ так, что $h(G_{|H|,d_H}) \geq p_H$ и добавим его ребра между вершинами графа H (значение p_H будет определено позже).

- Для каждого $i \in \{1, 2, \ldots, Z\}$, используя Теорему 3.1, мы построим d_{H_i}-регулярный экспандер $G_{|H_i|,d_{H_i}}$ удовлетворяющий неравенству $h(G_{|H_i|,d_{H_i}}) \geq p_{H_i}$ и добавим его ребра между вершинами графа H_i(точное значение p_{H_i} будет определено позже).

- Для каждого $i \in \{1, 2, \ldots, Z\}$ мы построим двудольный граф с долями $V(G)$ и $V(H_i)$ так, что все вершины $V(G)$ в этом двудольном графе имеют степень 1, а степени вершин из множества $V(H_i)$ отличаются между собой не больше, чем на единицу. Ребра всех таких двудольных графов добавим в граф G'. Обозначим максимальную степень вершин в доли $V(H_i)$ через $\Delta_{H,G}$, отметим, что $\Delta_{H,G} \leq \lceil \frac{1}{\phi} \rceil$.

Обратим внимание, что мы строим мультиграф, то есть, когда ребро добавляется несколько раз, то создаются его копии.

Покажем, что в оптимальном линейном упорядочивании графа G' выполняется следующее:

1. Вершины подграфа H расположены последовательно.

2. Для любого $i \in \{1, 2, \ldots Z\}$ вершины подграфа H_i расположены последовательно.

3. В упорядочивании вершины подграфа H делят вершины G на две примерно равные части (вершины до H и после).

Отметим, что свойство 2 играет важную роль в доказательстве свойства 3.

Напомним, что конструкция графа G' зависит не только от графа G, но также и от параметров Z, ϕ, p_H, p_{H_i}, которые влияют на значения d_H и d_{H_i} по теореме 3.1.

Ниже в этом разделе мы считаем, что вершины графа G' упорядочены слева направо согласно некоторому упорядочиванию π. Вершину, отображенную перестановкой π в i, будем называть i-ой вершиной (слева). Множество вершин U *последовательно в* π, если $\pi(U) = \{p, p+1, \dots q-1, q\}$ для некоторых $p, q \in \mathbf{N}$. Множество всех вершин, расположенных в перестановке π перед множеством (слева от множества) U, будем обозначать $L(U)$. Аналогично обозначим множество всех вершин после (справа от) U через $R(U)$. *Блоком* множества U назовём произвольное максимальное по включению подмножество U, чьи вершины расположены в перестановке π последовательно. *Левым блоком* U назовём блок множества U, отображённый перестановкой π в наименьшие значения среди $\pi(U)$. *Вторым левым* блоком U назовём самый первый блок U, расположенный справа от левого блока множества U. Множество вершин из множества $V(G') \setminus U$, расположенных между левым и вторым левым блоком, назовём *внутренним блоком* множества U. Если U состоит из одного блока, то внутренний блок не существует.

В дальнейшем мы часто будем пользоваться *перестановочной леммой.* Лемма описывает, при каких условиях изменение порядка двух множеств приводит к уменьшению стоимости упорядочивания.

Лемма 3.5 (Перестановочная лемма)**.** Рассмотрим перестановку π произвольного графа G'. Предположим, вершины множеств $X, Y, X \cup Y \subseteq V(G')$ расположены последовательно и вершины множества X расположены до Y. Положим $L := L(X)$ и $R := R(Y)$. Пусть

- P_X — верхняя граница на степени вершин из множества X в двудольном индуцированном подграфе $G'[L, X]$.
- P_C — верхняя граница на степени вершин в $G'[X, Y]$.
- P_Y — верхняя граница на степени вершин из множества Y в графе $G'[Y, R]$.

И пусть p — это нижняя оценка на среднюю степень вершины из X в графе $G'[X, R]$. Тогда из неравенства $p > P_X + 2P_C + P_Y$ следует, что изменение порядка следования (X, Y) на (Y, X) влечёт к уменьшению стоимости упорядочивания π.

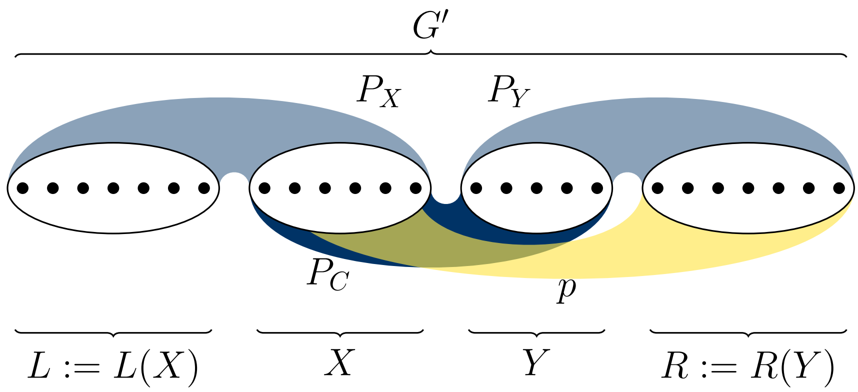


Рис. 3.3: В лемме 3.5 множество вершин графа G' разделено на четыре множества L, X, Y и R. Значения P_X, P_Y, P_C (верхние оценки) и p (нижняя оценка) на степени вершин в индуцированных двудольных графах показаны оттенком синего и жёлтого цветов.

Доказательство. Расположение множеств L, X, Y, R и оценки на степени их вершин показаны на Рис. 3.3. Длина ребра, полностью лежащего в одном из множеств L, X, Y, R, остаётся неизменной после изменения порядка следования множеств X и Y. Также не изменяется длина рёбер, идущих из L в R. Длина каждого ребра, идущего из X в Y, возрастает не более чем на $|X| + |Y| \leq 2\max\{|X|, |Y|\}$. Стоимость рёбер, соединяющих вершины X с вершинами из L, возрастает на $|Y|$. Аналогично стоимость рёбер, идущих из Y в R, возрастает ровно на $|X|$. С другой стороны, рёбра, соединяющие вершины множества X с вершинами множества R, стали короче на $|Y|$. Верхние оценки на максимальную степень и нижние оценки на среднюю степень из условия леммы позволяют ограничить снизу изменение стоимости упорядочивания после перестановки. Так, общее уменьшение стоимости рёбер соединяющих X и R не меньше $p|X||Y|$. Таким образом, уменьшение стоимости после перестановки составляет не менее

$$p|X||Y| - 2\min\left\{|X|, |Y|\right\} P_C \max\left\{|X|, |Y|\right\} - |X|P_X|Y| - |Y|P_Y|X|,$$

что равняется

$$|X||Y|(p - 2P_C - P_X - P_Y).$$

Учитывая неравенство из условия леммы, получаем, что значение выражения меньше нуля, а, значит, стоимость упорядочивания уменьшилась. $\square$

Докажем несколько утверждений о том, как выглядит оптимальное упорядочивание π графа $G' := T(G)$, где G — это d_G-регулярный граф. Напомним, что $Z, \varphi, p_H, p_{H_i}, d_H$ и d_{H_i} — это параметры преобразования $T(\cdot)$. Значения этих параметров будут выбраны позже.

Лемма 3.6. Если $p_H > 3\Delta_{H,G} + 3Z + d_G$, а π — оптимальное линейное упорядочивание графа G', то вершины $V(H)$ расположены последовательно в π.

Доказательство. Предположим противное, то есть вершины $V(H)$ расположены не последовательно в π. Рассмотрим левый блок $V(H)$ и обозначим его элементы через X. Предположим, что $|X| \leq \frac{|H|}{2}$, иначе вместо самого левого блока мы рассмотрим самый правый блок $V(H)$ и применим симметричные рассуждения. Обозначим через Y внутренний блок $V(H)$ и положим $L := L(X), R := R(Y)$. Следующие значения удовлетворяют условиям леммы 3.5:

$$P_X := \Delta_{H,G}, \qquad P_Y := d_G + Z, \qquad P_C := \Delta_{H,G} + Z.$$

Поскольку H — экспандер, $|X| \leq |H|/2$ и $H \setminus X \subseteq R$, мы можем положить $p = p_H$. Осталось доказать справедливость неравенства из условия перестановочной леммы. Имеем:

$$p = p_H > 3\Delta_{H,G} + 3Z + d_G = P_X + 2P_C + P_Y.$$

Таким образом, мы можем поменять X и Y местами и уменьшить стоимость упорядочивания, что противоречит оптимальности π. $\square$

Лемма 3.7. Пусть $i \in \{1, \ldots, Z\}$. Если $p_{H_i} > d_{H_{i+1}} + 4d_H + 2\Delta_{H,G}$, π — оптимальное линейное упорядочивание G' и для любого $i' < i$ вершины графа $H_{i'}$ расположены последовательно в π, тогда вершины H_i тоже расположены последовательно в упорядочивании π. (Для удобства обозначений на протяжении всей леммы мы будем считать, что $d_{H_{Z+1}} := 0$.)

Доказательство. Предположим противное. Пусть вершины H_i расположены непоследовательно. Обозначим через X левый блок $V(H_i)$ в упорядочивании π. Аналогично, как и в лемме 3.6, предположим, что $|X| \leq |H_i|/2$ (иначе мы рассмотрим самый правый блок вместо левого и применим аналогичные рассуждения). Покажем, что перемещение X вправо уменьшает стоимость упорядочивания.

Обозначим через u вершину, расположенную в упорядочивании π непосредственно справа от X. Согласно лемме 3.6 известно, что $u \notin V(G)$. А значит, $u \in V(H_j)$ для $j \neq i$. Рассмотрим два случая $j < i$ и $j > i$.

Предположим, что $u \in V(H_j)$ для $j < i$. Заметим, что вершины H_j расположены последовательно в π. Положим $Y := V(H_j), L := L(X)$, и $R := R(Y)$. Снова воспользуемся перестановочной леммой. Возьмём следующие значения верхних оценок:

$$P_X := \Delta_{H,G} + d_H, \qquad P_Y := \Delta_{H,G} + d_H, \qquad P_C := d_H.$$

Поскольку H_i — экспандер, $|X| \leq |H_i|/2$ и $V(H_i) \setminus X \subseteq R$, мы можем положить нижнюю оценку на среднее значение степени p равной p_{H_i}. По условию данной леммы мы имеем:

$$p = p_{H_i} > 4d_H + 2\Delta_{H,G} = P_X + 2P_C + P_Y.$$

Таким образом, неравенство из перестановочной леммы выполняется и мы можем уменьшить стоимость упорядочивания π в этом случае.

Предположим, что $u \in V(H_j)$ для $j > i$. Мы снова воспользуемся леммой 3.5, чтобы передвинуть блок X на одну позицию вправо, поменяв при этом местами X и $Y := \{u\}$. Положим $L := L(X), R := R(Y)$ и:

$$P_X := \Delta_{H,G} + d_H, \qquad P_Y := \Delta_{H,G} + d_H + d_{H_{i+1}}, \qquad P_C := d_H.$$

Аналогично предыдущему случаю, положим $p := p_{H_i}$. Неравенство из перестановочной леммы снова выполнено, поскольку:

$$p = p_{H_i} > 4d_H + 2\Delta_{H,G} + d_{H_{i+1}} = P_X + 2P_C + P_Y.$$

Тем самым получено противоречие с оптимальностью упорядочивания π.

$\square$

Согласно лемме 3.6 мы знаем, что оптимальное линейное упорядочивание графа G' располагает вершины H последовательно, если выполнено неравенство из условия леммы. Более того, итеративно применив лемму 3.7, мы получаем, что для любого i вершины множества H_i расположены последовательно. Другими словами, в оптимальном линейном упорядочивании вершины H расположены в порядке $H_{\ell_1}, H_{\ell_2}, \ldots, H_{\ell_Z}$, где $(\ell_1, \ell_2, \ldots, \ell_Z)$ — некоторая перестановка $\{1, 2, \ldots, Z\}$. Вершины графа G могут расположиться только слева или справа от вершин H. Следующая лемма показывает, что H делит граф G на примерно две равные части.

Лемма 3.8. Пусть граф G' получен с помощью преобразования $T(\cdot)$. При этом параметры преобразования удовлетворяют неравенствам из лемм 3.6 и 3.7, а также неравенству $Z\varphi \geq 2$. Рассмотрим оптимальное линейное упорядочивание π графа G', положим $A := L(H), B := R(H)$ и $\gamma = 3\varphi d_G$. Тогда справедливо следующее неравенство $\big||A| - |B|\big| \leq \gamma n$.

Доказательство. Предположим дисбаланс $\big||A| - |B|\big|$ строго больше γn. Не ограничивая общности, предположим, что $|A| > |B|$. Рассмотрим самую левую вершину u, то есть такую, что $\pi(u) = 1$.

Подсчитаем изменение стоимости упорядочивания, если мы переместим вершину u в самый конец упорядочивания. Общая стоимость рёбер внутри G может увеличиться не более чем на $d_G(Z\lceil\varphi n\rceil + n) \leq d_G(Z\varphi n + Z + n)$. Заметим, что для достаточно больших n мы имеем $\frac{Z\varphi n}{2} \geq Z$, и по предположению леммы $\frac{Z\varphi n}{2} \geq n$, значит общая стоимость рёбер из G увеличилась не более, чем на $d_G(Z\varphi n + Z + n) \leq 2d_G Z\varphi n$.

Кроме этого вершина u соединена ровно с одной вершиной v_i из H_{ℓ_i}. До перемещения вершины u стоимость ребра uv_i была $|A| + (i-1)\lceil\varphi n\rceil + r_i$ для некоторого $0 \leq r_i \leq \lceil\varphi n\rceil - 1$, а после перемещения она стала

$|B| + ((Z - i + 1)\lceil\varphi n\rceil - r_i)$, таким образом, вклад в стоимость таких рёбер изменился на:

$$(|B| - |A|)Z + \sum_{i=1}^{Z}((Z - i + 1)\lceil\varphi n\rceil - r_i) - \sum_{i=1}^{Z}((i - 1)\lceil\varphi n\rceil + r_i) =$$

$$(|B| - |A|)Z + \sum_{i=1}^{Z}(\lceil\varphi n\rceil - 2r_i) <$$

$$-\gamma n Z + Z\lceil\varphi n\rceil \overset{\gamma=3d_G\varphi}{=} -3d_G Z\varphi n + Z\lceil\varphi n\rceil \overset{Z\lceil\varphi n\rceil < d_G Z\varphi n}{<} -2d_G Z\varphi n$$

Отметим, что в последнем неравенстве мы воспользовались предположением $\lceil\varphi n\rceil < d_G\varphi n$. Это предположение верно для $d_G \geq 2$ и достаточно большого n, поскольку φ — константа.

Учитывая, что общее увеличение стоимости рёбер G не превосходит $2d_G Z\varphi n$, мы получаем, что перемещение u в конец перестановки уменьшает стоимость упорядочивания, что противоречит оптимальности упорядочивания π. □

Перейдём к доказательству теоремы 3.8.

Доказательство теоремы 3.8. Докажем теорему, построив сведение по Тьюрингу от задачи о минимальной бисекции с зазором$(d)_{[\alpha,\beta]}$ к задаче об оптимальном линейном упорядочивании ОЛУ$_{\leq}(d')$, где

$$d > \frac{4}{\beta - \alpha} \tag{3.2}$$

и d' — некоторая константа, зависящая от d, α, и β. Сведение устроено следующим образом.

Для данного экземпляра G задачи о минимальной бисекции$(d)_{[\alpha,\beta]}$ построим эквивалентный экземпляр (G', k) задачи об оптимальном линейном упорядочивании(d'). Для этого применим преобразование $T(\cdot)$ к графу G. Значение параметра k из задачи об оптимальном линейном упорядочивании будет зависеть не только от параметров преобразования $T(\cdot)$, но и от оптимального упорядочивания экспандера H (напомним, что H индуцированный подграф G'). Как было отмечено ранее, наше преобразование является сведением по Тьюрингу. А значит, имеет доступ к оракулу, который может вычислить стоимость оптимального линейного упорядочивания.

Значения параметров преобразования будут выбраны в следующем порядке.

- Сперва положим $\gamma := \frac{\beta-\alpha}{4}$ и $\varphi := \frac{\gamma}{3d_G}$. Заметим, что $\gamma, \varphi \in (0,1)$ и γ, φ удовлетворяют условию $\gamma = 3\varphi d_G$ из леммы 3.8.

- Далее возьмём значение Z равное:

$$Z := \lceil \frac{2(2\alpha+1)}{(\beta-\alpha)\varphi}) \rceil\,.$$

  В частности, $Z \geq \frac{2}{\varphi}$, а, значит, удовлетворяет условию леммы 3.8. Более того, также верно следующее неравенство:

$$2(2\alpha+1) \leq (\beta-\alpha)Z\varphi. \tag{3.3}$$

- По построению $\Delta_{H,G} \leq \lceil \frac{1}{\varphi} \rceil = \lceil \frac{3d_G}{\gamma} \rceil$.

- Зададим значение p_H, которое определяет значение d_H по теореме 3.1:

$$p_H := 3\Delta_{H,G} + 3Z + d_G + 1\,.$$

  Мы прибавляем 1 исключительно для того, чтобы неравенство из леммы 3.6 было строгим. Вместо единицы можно использовать любую другую положительную константу.

- Наконец, определим значение p_{H_i},

$$p_{H_i} := d_{H_{i+1}} + 4d_H + 2\Delta_{H,G} + 1,$$

  в убывающем порядке значений индекса $i = Z, \ldots, 1$. Для удобства обозначений считаем, что $d_{H_{Z+1}} = 0$. Отметим, что значение d_{H_i} определяется значением p_{H_i} с помощью теоремы 3.1.

Заметим, что подобным образом выбранные значения констант удовлетворяют всем требованиям лемм 3.6, 3.7, 3.8.

Пусть G' — это результат преобразования $T(G)$ со значением параметров, указанными выше. Значение k мы установим равным:

$$k := \mathrm{OLA}(H) + \alpha m \cdot (Z\lceil\varphi n\rceil + n) + m \cdot \frac{n}{2} + \left((\frac{n}{2}+1)\frac{n}{2}Z + n\sum_{i=1}^{Z} i\lceil\varphi n\rceil\right). \tag{3.4}$$

Заметим, что для вычисления значения k мы пользуемся вызовом оракула для вычисления стоимости оптимального линейного упорядочивания графа H. Осталось показать, что размер бисекции G не превосходит αm тогда и только тогда, когда стоимость оптимального линейного упорядочивания G' не более k.

Предположим, что бисекция графа G содержит не более αm рёбер. Мы утверждаем, что k является оценкой сверху на стоимость оптимального линейного упорядочивания графа G'. Покажем, как, имея разбиение графа G с минимальной бисекцией, построить упорядочивание G' со стоимостью, не превосходящей k. Предположим, значение бисекции достигает своего минимума, когда мы разделяем вершины $V(G)$ на два множества A и B. Рассмотрим упорядочивание π, в котором в начале стоят только вершины из множества A, затем вершины графа H в порядке, при котором достигается значение оптимального упорядочивания графа H, и в самом конце стоят вершины из множества B.

Первое слагаемое выражения (3.4) соответствует стоимости всех рёбер внутри графа H. Стоимость рёбер, соединяющих вершины A и B, не превосходит второго слагаемого, ведь количество таких рёбер не больше αm, а стоимость одного ребра не превосходит $|V(G')| = (Z\lceil\varphi n\rceil + n)$. Третье слагаемое ограничивает стоимость рёбер, находящихся внутри множеств A или B. Ведь таких рёбер не больше m и стоимость каждого ребра не более $\frac{n}{2}$, поскольку $|A| = |B| = \frac{n}{2}$. Последнее слагаемое ограничивает стоимость рёбер, соединяющих вершины G и H. Из каждой вершины v графа G выходит ровно одно ребро к какой-то вершине подграфа H_{ℓ_i}. Если $v \in A$, то мы можем ограничить стоимость такого ребра значением $j(v) + i\lceil\varphi n\rceil$, где $j(v)$ это расстояние в упорядочивании от вершины v до самой первой вершины H. Вначале мы посчитаем вклад $j(v)$-слагаемых в вышеописанном выражении. Подсчёт произведём по всем $v \in A$. Поскольку $|A| = n/2$, то, суммируя по всем $v \in A$ и $i = 1, \ldots, Z$ мы получим $\sum_{j=1}^{|A|} jZ = \frac{(\frac{n}{2}+1)\frac{n}{2}}{2}Z$. Ситуация аналогична с множеством B. Последнее слагаемое выражения (3.4) получается суммированием оставшейся стоимости рёбер: $i\lceil\varphi n\rceil$ для всех $i = 1, \ldots, Z$ и $v \in G$. Комбинируя все вместе, получаем, что k является оценкой сверху на стоимость оптимального линейного упорядочивания.

Докажем утверждение в другую сторону. Предположим, что граф G' имеет оптимальное упорядочивание π стоимостью не больше k. Нам требуется доказать, что в графе G размер бисекции не превосходит αm. Леммы 3.6, 3.7, 3.8 гарантируют определённую структуру упорядочивания π. В частности, вершины подграфа H должны быть расположены последовательно в упорядочивании π. Воспользуемся данным фактом для построения бисекции графа G. Если $|A| < |B|$, то поменяем местами A и B. Положим $A := L(H)$, $B := R(H)$. Отметим, что (A, B) — это разбиение множества вершин $V(G)$. Более того, разница размеров множеств A и B ограничена по лемме 3.8. Ограничим число рёбер между A и B. Для этого ограничим снизу стоимость упорядочивания π через $|E_{G'}(A, B)|$. Стоимость составляет не менее:

$$\mathrm{OLA}(H) + |E_{G'}(A, B)| \cdot Z\lceil\varphi n\rceil + \Big((\frac{n}{2} + 1)\frac{n}{2}Z + n\sum_{i=1}^{Z}(i-1)\lceil\varphi n\rceil\Big). \tag{3.5}$$

Здесь первое слагаемое отвечает за стоимость рёбер подграфа H, поскольку оно ограничено снизу стоимостью оптимального линейного упорядочивания графа H. Второе слагаемое выражения (3.5) — это нижняя оценка на стоимость рёбер G, идущих из A в B. Количество таких рёбер равняется $|E_{G'}(A, B)|$, и стоимость каждого ребра составляет не менее $Z\lceil\varphi n\rceil$. Напомним, что вершины H имеют следующий порядок

$H_{\ell'_1}, H_{\ell'_2}, \ldots, H_{\ell'_Z}$, где $(\ell'_1, \ell'_2, \ldots, \ell'_Z)$ некоторая перестановка $\{1, 2, \ldots, Z\}$. Третье слагаемое ограничивает снизу стоимость рёбер, соединяющих вершины G с вершинами H. Любая вершина v из G имеет ровно одно ребро, идущее в $H_{\ell'_i}$. Аналогично анализу, представленному выше, мы ограничим стоимость такого ребра выражением $j(v) + (i-1)\lceil\varphi n\rceil$. Суммарный вклад $j(v)$ по всем вершинам $v \in G$ равняется $\left(\sum_{j=1}^{|A|} j + \sum_{j=1}^{|B|} j\right) Z \geq \left(2\sum_{j=1}^{\frac{n}{2}} j\right) Z \geq (\frac{n}{2}+1)\frac{n}{2} Z$. Оставшаяся часть получается суммированием $(i-1)\lceil\varphi n\rceil$ по всем $v \in G$ и $i = 1, \ldots, Z$. Сравнивая (3.4) с (3.5), мы получим:

$$|E_{G'}(A,B)| \cdot Z\lceil\varphi n\rceil \leq \alpha m \cdot (Z\lceil\varphi n\rceil + n) + m \cdot \frac{n}{2} + n\sum_{i=1}^{Z} i\lceil\varphi n\rceil - n\sum_{i=1}^{Z}(i-1)\lceil\varphi n\rceil =$$

$$\alpha m \cdot Z\lceil\varphi n\rceil + \frac{2(2\alpha+1)mn}{4} + n\sum_{i=1}^{Z}\lceil\varphi n\rceil \overset{(3.3)}{\leq}$$

$$\alpha m \cdot Z\lceil\varphi n\rceil + \frac{(\beta-\alpha)m \cdot Z\varphi n}{4} + n \cdot Z\lceil\varphi n\rceil \overset{\varphi n \leq \lceil\varphi n\rceil}{\leq}$$

$$\alpha m \cdot Z\lceil\varphi n\rceil + \frac{(\beta-\alpha)m}{4} \cdot Z\lceil\varphi n\rceil + n \cdot Z\lceil\varphi n\rceil \overset{n=\frac{2}{d_G}m<\frac{\beta-\alpha}{2}m \quad (3.2)}{<} \frac{\alpha+3\beta}{4} m \cdot Z\lceil\varphi n\rceil.$$

Таким образом, получается $|E_{G'}(A,B)| < \frac{\alpha+3\beta}{4}m$. Создадим бисекцию (A', B') графа G следующим образом. Пусть C это произвольное подмножество A, состоящее из $\frac{|A|-|B|}{2}$ вершин. Положим $A' = A - C$, $B' = B \cup C$. Переместив вершины C, мы получим не более $\frac{\gamma n}{2} \cdot d_G = \gamma m$ новых рёбер в разрезе, а значит:

$$|E_{G'}(A', B')| \leq |E_{G'}(A,B)| + \gamma m < \frac{\alpha+3\beta}{4}m + \frac{\beta-\alpha}{4}m = \beta m.$$

Таким образом, $|E_{G'}(A', B')| < \beta m$. Поскольку рассматривается задача с гарантированным зазором, то известно, что или G имеет бисекцию размера меньше αm, или не существует бисекции меньше βm. Поэтому мы заключаем, что в нашем случае размер бисекции не превосходит αm.

$\square$

3.3 Нижние оценки на вычислительную сложность задачи о рёберных дополнениях

В этом разделе докажем теоремы 3.9 и 3.10, таким образом, будут доказаны нижние оценки на основании гипотезы экспоненциального времени и гипотезе 3.2. Данные нижние оценки будут справедливы для задач о цепочном дополнении, хордальном дополнении, интервальном дополнении, собственно интервальном дополнении, тривиально совершенном дополнении и пороговом дополнении. В качестве отправной точки в наших доказательствах будем использовать теоремы 3.2 и 3.8. Таким образом, наша цель заключается в том, чтобы преобразовать экземпляр задачи об оптимальном линейном упорядочивании в экземпляр задачи о рёберном дополнении. Главное преобразование данного раздела трансформирует экземпляр задачи об оптимальном линейном упорядочивании в экземпляр задачи о цепочном дополнении. Наше преобразование является модификацией преобразования Янакакиса [57]. Единственным отличием нашего преобразования является то, что при трансформации экземпляра с ограниченной степенью мы получим линейное число вершин в конечном экземпляре задачи о цепочном дополнении. Данный факт играет важную роль для доказательства теоремы 3.10, в то время как для доказательства теоремы 3.9 можно воспользоваться оригинальным сведением Янакакиса.

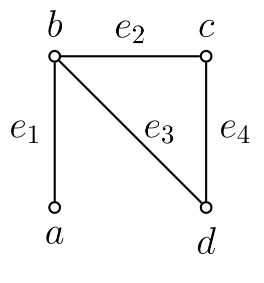
Теорема 3.9. Если гипотеза ETH верна, тогда существует такое целое $c \geq 1$, что задачи хордального, интервального, собственно интервального, порогового, тривиально совершенного, цепочного дополнений не допускают алгоритма с временем работы $2^{\mathcal{O}(\sqrt{n}/\log^c n)}$, а, значит, и с временем $2^{\mathcal{O}(k^{1/4}/\log^c k)} \cdot n^{\mathcal{O}(1)}$.

Теорема 3.10. Если гипотеза 3.2 верна, тогда не существует алгоритма для задачи цепочного дополнения с временем работы $2^{o(n+m)}$, а для задач хордального, интервального, собственно интервального, порогового, тривиально совершенного дополнений не существует алгоритма с временем работы $2^{o(n)}$. Таким образом, ни одна их этих задач не может быть решена за время $2^{o(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$.

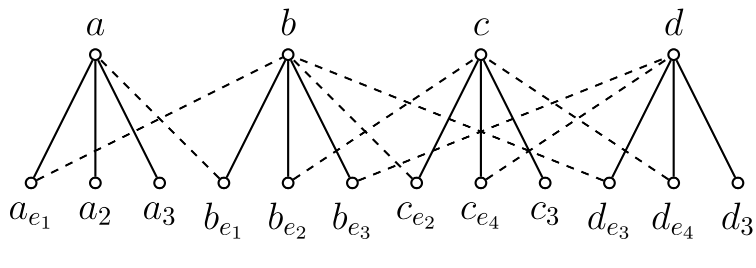
Определение 3.1. Двудольный граф (A, B, F) с вершинами $A \uplus B$ и рёбрами F будем называть *цепочным графом*, если множество вершин A (называемых левой стороной) может быть упорядочено $v_1, v_2, \ldots, v_n$ (такой порядок будем называть левым порядком) так, что $N(v_1) \subseteq N(v_2) \subseteq \cdots \subseteq N(v_n)$.

В задаче о дополнении до цепочного графа (задача цепочного дополнения) нам дан двудольный граф (A, B, F) и требуется найти такое множество $F' \subseteq A \times B \setminus F$ минимального размера, что двудольный граф $(A, B, F \cup F')$ является цепочным графом.

Лемма 3.9. Существует полиномиальный алгоритм, принимающий на вход экземпляр задачи об оптимальном упорядочивании $I = (G = (V, E), k)$ и выдающий эквивалентный экземпляр $I' = (G' = (A, B, F), k')$ задачи о цепочном дополнении. При этом число вершин в графе G' ограничено величиной $\mathcal{O}(\Delta_G \cdot |V|)$, где Δ_G — это максимальная степень вершины в графе G. Отметим, что преобразование работает, даже если граф G содержит кратные ребра G.



(a) G экземпляр задачи ОЛУ



(b) G' экземпляр цепочного дополнения

Рис. 3.4: Преобразование задачи об оптимальном линейном упорядочивании в задачу цепочного дополнения.

Доказательство. В качестве левой стороны G' возьмём $A = V$. Для каждой вершины $v \in V$ создадим множество S_v, содержащее Δ_G новых вершин $S_v = \{v_e : e \in \delta_G(v)\} \cup \{v_i : \deg(v) < i \leq \Delta_G\}$. В качестве B возьмём объединение всех множеств S_v, то есть B содержит в точности $\Delta_G \cdot |V|$ вершин. Множество рёбер F будет построено следующим образом. Для каждого $w \in S_v$ добавим к F ребро vw. Кроме этого, для каждой вершины $v_e \in B$, где $e \in E, e = uv$, добавим в множество F ребро uv_e. В результате степени вершин v_e в графе G' будут равны двум. Пример описанного преобразования изображён на Рис. 3.4. Для завершения сведения достаточно положить $k' = k + \Delta_G \frac{n(n-1)}{2} - 2|E|$.

Для упорядочивания π вершин графа $G = (V, E)$ обозначим через $C(G, \pi)$ стоимость упорядочивания π (то есть $C(\pi, G) = \sum_{uv \in E} |\pi(u) - \pi(v)|$). Для упорядочивания σ левой стороны A двудольного графа $G' = (A, B, F)$ обозначим через $E(G', \sigma)$ число рёбер, которые необходимо добавить, чтобы получить цепочный граф, у которого левый порядок совпадает с σ. Эквивалентность экземпляров I и I' следует из следующего утверждения:

Утверждение 3.11. Для любого упорядочивания π множества V (или множества A) верно равенство $E(G', \pi) = C(G, \pi) + \Delta_G \frac{n(n-1)}{2} - 2|E|$.

Доказательство. Вначале мы определим число рёбер в минимальном (относительно рёберного включения) цепочном двудольном графе G'', в котором левый порядок равен $\pi = (v_1, v_2, \ldots, v_n)$ и граф $G' = (A, B, F)$ является подграфом G''. Мы знаем, что $N(v_i) \subseteq N(v_j)$ для любого $i < j$, следовательно любая вершина $x \in S_{v_i}$ должна быть соединена ребром со всеми вершинами, стоящими на позициях $\{v_i, v_{i+1}, \ldots, v_n\}$. Это означает, что любая вершина из S_{v_i} соединена как минимум с $(n+1) - i$ вершинами из A. Более того, из минимальности G'' следует, что вершина x соединена в точности с таким количеством вершин, если x не соответствует никакому ребру из G, другими словами, если степень вершины x равна единице в G'. Если вершина $w_i \in S_{v_i}$ соответствует некоторому ребру $e \in G$, которое инцидентно вершинам v_i, v_j (напомним, что мы работаем с мультиграфами и поэтому несколько различных рёбер могут быть инцидентны одним и теми же вершинам), в таком случае вершина w_i соединена с вершинами v_i, v_j в графе G'. А, значит, в минимальном цепочном графе G'' степень вершины w_i равна или $(n+1)-i$, или $(n+1)-j = (n+1)-i+(i-j)$, в зависимости от того, $i < j$ или $i > j$. Заметим, что есть вторая вершина $w_j \in S_{v_j}$, которая тоже соответствует ребру e. Степени вершин w_i и w_j в G'' равняются $(n+1)-i$ или $(n+1)-j$, в зависимости от того $i < j$ или $i > j$. В обоих случаях сумма степеней w_i, w_j в G'' может быть записана как $((n+1)-i) + ((n+1)-j) + |i-j|$. Таким образом, для каждого ребра e, инцидентного вершинам v_i, v_j, у нас возникает дополнительно $|i-j|$ рёбер. Суммируя, мы получаем, что число рёбер в графе G'' равняется:

$$\Delta_G(\sum_i^n((n+1)-i)) + \sum_{v_iv_j \in E}|i-j| = \Delta_G\frac{n(n+1)}{2} + C(G,\pi).$$

Число добавленных рёбер к графу G' равняется числу рёбер в G'' минус число рёбер в G'. Таким образом, мы добавили ровно

$$(\Delta_G\frac{n(n+1)}{2} + C(G,\pi)) - (\Delta_G n + 2|E|) = \Delta_G\frac{n(n-1)}{2} + C(G,\pi) - 2|E|$$

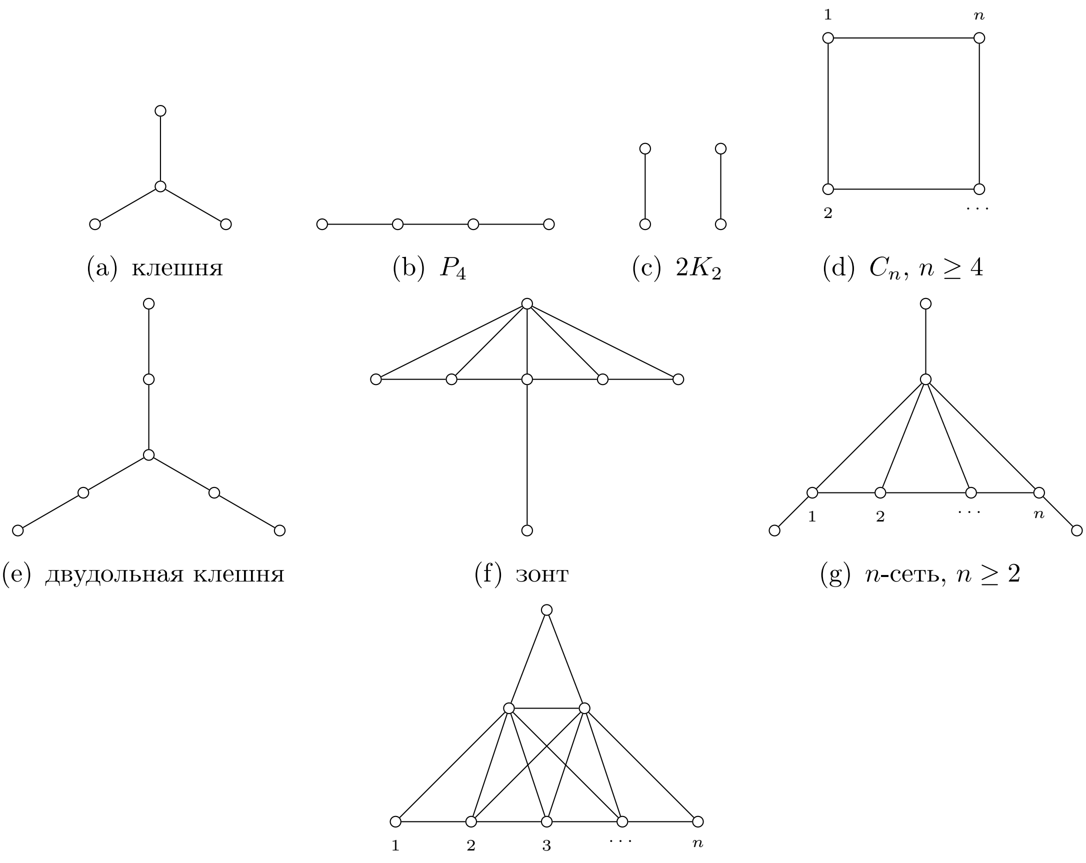
рёбер. □

Доказательство утверждения завершает доказательство леммы.

□

Классы хордальных, интервальных, собственных интервальных, тривиально совершенных и пороговых графов допускают несколько различных характеризаций. Однако, для наших целей удобнее всего использовать характеризацию с помощью запрещённых индуцированных подграфов. Характеризация представлена в таблице 3.1.

Лемма 3.10. Существуют полиномиальные по времени алгоритмы, трансформирующие задачу цепочного дополнения в задачи о рёберных дополнениях до хордального, интервального, собственно интервального, порогового и тривиально совершенного графов. Более того, приведённые алгоритмы выдают граф на том же множестве вершин, что и исходный граф.



(a) клешня (b) P_4 (c) $2K_2$ (d) C_n, $n \geq 4$

(e) двудольная клешня (f) зонт (g) n-сеть, $n \geq 2$

(h) n-палатка, $n \geq 3$

Рис. 3.5: Запрещённые индуцированные подграфы для различных классов графов.

Доказательство. Для любого двудольного графа $H' = (U_1, U_2, F)$ мы рассмотрим граф $Ch(H') = (U_1, U_2, F \cup \{uv | u, v \in U_1\} \cup \{uv | u, v \in U_2\})$. Янакакис в работе [57] показал, что любой двудольный граф H' является цепочным графом тогда и только тогда, когда $Ch(H')$ — это хордальный граф. Следовательно, для того, чтобы трансформировать экземпляр задачи о цепочном дополнении $H = (A, B, F)$ в экземпляр задачи о хордальном дополнении, достаточно построить клики на множестве вершин A и B. Таким образом, полученный граф представляет из себя объединение двух клик, между которыми проведены некоторые ребра. Получается, что к выданному графу нельзя добавить какие-то ребра так, чтобы в нем содержался один из следующих графов: клешня, двудольная клешня, зонт, p-сеть, q-палатка для любых $p \geq 2, q \geq 3$. Это утверждение вер-

Название класса графов	Запрещённые индуцированные подграфы
Хордальные	C_n для $n \geq 4$
Собственно интервальные	клешня, 2-сеть, 3-палатка, C_n для $n \geq 4$
Интервальные	двудольная клешня, зонт, n-сеть для $n \geq 2$, n-палатка для $n \geq 3$, C_n для $n \geq 4$
Пороговые	$2K_2, C_4, P_4$
Тривиально совершенные	C_4, P_4

Таблица 3.1: Характеризация классов графов с помощью запрещённых индуцированных подграфов.

но, поскольку все эти графы содержат независимое множество из трёх вершин, а объединение двух клик — нет. Следовательно, решения для задач хордального, интервального и собственно интервального дополнений совпадают на подобных экземплярах (объединение двух клик и возможно ещё некоторые ребра). А значит, сведение к задаче хордального дополнения можно рассматривать как сведение к задаче интервального или собственно интервального дополнения.

Для завершения доказательства леммы осталось привести сведение задачи цепочного дополнения к задачам о дополнении до порогового и тривиально совершенного графов. Если в задаче цепочного дополнения нам дан на вход двудольный граф $H = (A, B, F)$, то мы рассмотрим задачи о дополнении до тривиально совершенного и порогового графов на графе $G = (A \cup B, F \cup \{(u, v)|u, v \in A\})$. То есть просто добавим несколько рёбер так, чтобы A стало кликой. Покажем, что минимальное цепочное дополнение графа H соответствует дополнению нового графа до тривиально совершенного или порогового графов. Пусть множество рёбер F' — это решение для задачи цепочного дополнения графа H. Рассмотрим граф $G' = (A \cup B, F \cup \{(u, v)|u, v \in A\} \cup F')$. G' — это объединение независимого множества и клики, где между кликой и независимым множеством проведены некоторые ребра. Такой граф не содержит индуцированных подграфов $2K_2$ или C_4. Покажем, что G' также не содержит

P_4. Если бы G' содержало индуцированный путь $P_4 = v_1v_2v_3v_4$, тогда вершины v_2, v_3 принадлежали бы клике, а вершины v_1, v_4 — независимому множеству. Однако, такое расположение противоречит тому, что ребра между кликой и независимым множеством формируют цепочный граф. Таким образом, граф G' не содержит индуцированных подграфов $2K_2, P_4, C_4$, а, значит, является тривиально совершенным и пороговым графом одновременно.

Докажем утверждение в обратную сторону. Пусть F' — решение для задачи о дополнении графа G' до тривиально совершенного или порогового графа . Сейчас нам достаточно показать, что $(A, B, (F \cup F') \cap E(A, B))$ является цепочным графом. Если данный граф не является цепочным графом, тогда он содержит два независимых ребра v_1v_2, v_3v_4 [57]. Однако, в таком случае вершины v_1, v_2, v_3, v_4 индуцируют путь P_4 в графе $G' \cup F'$, что противоречит тому, что $G' \cup F'$ пороговый или тривиально совершенный граф.

Отметим, что наши сведения от задачи цепочного дополнения к задачам о хордальном, интервальном, собственно интервальном, пороговом и тривиально совершенном дополнениям не изменяют множество вершин, а лишь добавляют некоторые ребра к графу. □

Мы почти доказали теорему 3.9 и теорему 4.

Теорема 3.9. *Если гипотеза ETH верна, то существует целое $c \geq 1$ такое, что задачи хордального, интервального, собственно интервального, порогового, тривиального совершенного и цепочного дополнений не допускают алгоритмов с временем работы $2^{\mathcal{O}(\sqrt{n}/\log^c n)}$, а, значит, и с временем $2^{\mathcal{O}(k^{1/4}/\log^c k)} \cdot n^{\mathcal{O}(1)}$.*

Доказательство. Предположим, утверждение теоремы неверно, тогда для одной из задач существует алгоритм с временем работы $2^{\mathcal{O}(\sqrt{n}/\log^c n)}$. Рассмотрим такую задачу. Если нам требуется решить задачу оптимального линейного упорядочивания на графе с n вершинами, мы можем просто свести её к рассматриваемой задаче. При этом, из леммы 3.9 и 3.10 следует, что в новой задаче граф будет состоять из $(\Delta_G + 1)n = O(n^2)$

вершин. Что влечёт наличие $2^{\mathcal{O}(\sqrt{n^2}/\log^c n^2)} = 2^{\mathcal{O}(n/log^c n)}$ алгоритма для задачи об оптимальном линейном упорядочивании. А это противоречит теореме 3.2. Поскольку $k \leq n^2$, то мы также получаем вторую нижнюю оценку $2^{\Omega(k^{1/4}/\log^c k)} \cdot n^{\mathcal{O}(1)}$ на время работы алгоритма. $\square$

Теорема 4. *Если гипотеза 3.2 верна, тогда для задач хордального, интервального, собственно интервального, порогового, тривиального совершенного и цепочного дополнений не существует алгоритма с временем работы $2^{o(n)}$. Таким образом, ни одна их этих задача не может быть решена за время $2^{o(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$.*

Доказательство. В разделе 3.2 мы показали, как преобразовать экземпляр задачи о минимальной бисекции на d-регулярном графе в эквивалентный экземпляр задачи об оптимальном линейном упорядочивании. Более того, выданный граф имеет ограниченную степень. После такого преобразования, применив лемму 3.9, мы получим сведение задачи о минимальной бисекции на d-регулярных графах к задаче о цепочном дополнении. При этом полученный экземпляр задачи цепочного дополнения будет содержать $\mathcal{O}(n)$ вершин и рёбер. А значит существование $2^{o(n+m)}$-алгоритма противоречит гипотезе 3.2. По лемме 3.10 мы можем свести задачу цепочного дополнения к задачам хордального, интервального, собственно интервального, тривиально совершенного и порогового дополнений, при этом не изменив множество вершин. Последовательно применяя все три сведения, получаем преобразование экземпляра с n вершинами задачи минимальной бисекции на регулярных графах в экземпляр задачи о дополнении до хордального, интервального, собственно интервального, тривиально совершенного или порогового графов. Более того, выданный экземпляр содержит $\mathcal{O}(n)$ вершин. Подобное сведение доказывает нижнюю оценку $2^{\Omega(n)}$ на время работы для любой задачи о дополнении, упомянутой выше. Также для этих задач мы получаем нижнюю оценку $2^{\Omega(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$, поскольку $k \leq n^2$. $\square$

Глава 4

Заключение

В работе приведён алгоритм для поиска наибольшего индуцированного хордального/интервального подграфа. Идеи, на которых построен алгоритм, могут быть использованы при построении аналогичных алгоритмов для других классов графов. Дальнейший интерес изучения представляет следующая гипотеза.

Гипотеза 1. *Для любого полиномиально распознаваемого класса* Π *со свойством наследственности задача максимального индуцированного подграфа может быть решена за время* c^n, *где* $c < 2$ — *константа.*

Также в работе построены доказательства нижних оценок на вычислительную сложность для задач рёберного дополнения в предположении гипотезы экспоненциального времени и более сильная нижняя оценка — в предположении сложности задачи минимальной бисекции. В связи с этим важным вопросом является связь гипотезы о минимальной бисекции с другими более изученными гипотезами, а также возможность ослаблений требований в сформулированной гипотезе.

Литература

[1] Bodlaender Hans L, de Fluiter Babette. On intervalizing k-colored graphs for DNA physical mapping // Discrete Applied Mathematics. — 1996. — Vol. 71, no. 1. — P. 55–77.

[2] Golumbic Martin Charles, Kaplan Haim, Shamir Ron. On the complexity of DNA physical mapping // Advances in Applied Mathematics. — 1994. — Vol. 15, no. 3. — P. 251–261.

[3] Parter Seymour. The use of linear graphs in Gauss elimination // SIAM review. — 1961. — Vol. 3, no. 2. — P. 119–130.

[4] Tarjan Robert E. Graph theory and Gaussian elimination. — Computer Science Department, School of Humanities and Sciences, Stanford University, 1975.

[5] Chung Fan RK, Mumford David. Chordal completions of planar graphs // Journal of Combinatorial Theory, Series B. — 1994. — Vol. 62, no. 1. — P. 96–106.

[6] Tarjan Robert E, Yannakakis Mihalis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs // SIAM Journal on computing. — 1984. — Vol. 13, no. 3. — P. 566–579.

[7] Fulkerson Delbert, Gross Oliver. Incidence matrices and interval graphs // Pacific journal of mathematics. — 1965. — Vol. 15, no. 3. — P. 835–855.

[8] Lewis John M, Yannakakis Mihalis. The node-deletion problem for hereditary properties is NP-complete // Journal of Computer and System Sciences. — 1980. — Vol. 20, no. 2. — P. 219–230.

[9] Yannakakis Mihalis. The effect of a connectivity requirement on the complexity of maximum subgraph problems // Journal of the ACM (JACM). — 1979. — Vol. 26, no. 4. — P. 618–630.

[10] Robson John Michael. Algorithms for maximum independent sets // Journal of Algorithms. — 1986. — Vol. 7, no. 3. — P. 425–440.

[11] Fomin Fedor V, Gaspers Serge, Pyatkin Artem, Razgon Igor. On the Minimum Feedback Vertex Set problem: exact and enumeration algorithms // Algorithmica. — 2008. — Vol. 52, no. 2. — P. 293–307.

[12] Raman Venkatesh, Saurabh Saket, Sikdar Somnath. Efficient exact algorithms through enumerating maximal independent sets and other techniques // Theory of Computing Systems. — 2007. — Vol. 41, no. 3. — P. 563–587.

[13] Fomin Fedor V., Todinca Ioan, Villanger Yngve. Exact Algorithm for the Maximum Induced Planar Subgraph Problem // Proceedings of the 19th Annual European Symposium on Algorithms, ESA 2011. — Vol. 6942 of Lecture Notes in Computer Science. — Springer, 2011. — P. 287–298.

[14] Pilipczuk Marcin, Pilipczuk Michał. Finding a Maximum Induced Degenerate Subgraph Faster Than 2^n // Proceedings of the 7th International Symposium on Parameterized and Exact Computation, IPEC 2012. — Vol. 7535 of Lecture Notes in Computer Science. — Springer, 2012. — P. 3–12.

[15] Fomin Fedor V, Gaspers Serge, Kratsch Dieter et al. Iterative compression and exact algorithms // Theoretical Computer Science. — 2010. — Vol. 411, no. 7. — P. 1045–1053.

[16] Gaspers Serge, Kratsch Dieter, Liedloff Mathieu. On independent sets and bicliques in graphs // Algorithmica. — 2012. — Vol. 62, no. 3-4. — P. 637–658.

[17] Fomin Fedor V., Villanger Yngve. Finding Induced Subgraphs via Minimal Triangulations // Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010. — Vol. 5 of LIPIcs. — Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. — P. 383–394.

[18] Gupta Sushmita, Raman Venkatesh, Saurabh Saket. Maximum r-regular induced subgraph problem: Fast exponential algorithms and combinatorial bounds // SIAM Journal on Discrete Mathematics. — 2012. — Vol. 26, no. 4. — P. 1758–1780.

[19] van't Hof Pim, Villanger Yngve. Proper interval vertex deletion // Algorithmica. — 2013. — Vol. 65, no. 4. — P. 845–867.

[20] Cao Yixin. Unit Interval Editing is Fixed-Parameter Tractable // arXiv preprint arXiv:1504.04470. — 2015.

[21] Hüffner Falk, Komusiewicz Christian, Moser Hannes, Niedermeier Rolf. Fixed-parameter algorithms for cluster vertex deletion // Theory of Computing Systems. — 2010. — Vol. 47, no. 1. — P. 196–217.

[22] Marx Dániel, Schlotter Ildikó. Obtaining a planar graph by vertex deletion // Algorithmica. — 2012. — Vol. 62, no. 3-4. — P. 807–822.

[23] Heggernes Pinar, van't Hof Pim, Jansen Bart MP et al. Parameterized complexity of vertex deletion into perfect graph classes // Fundamentals of Computation Theory / Springer. — 2011. — P. 240–251.

[24] Marx Dániel. Chordal deletion is fixed-parameter tractable // Algorithmica. — 2010. — Vol. 57, no. 4. — P. 747–768.

[25] Lund Carsten, Yannakakis Mihalis. The approximation of maximum subgraph problems // Automata, Languages and Programming / Ed. by Andrzej Lingas, Rolf Karlsson, Svante Carlsson. — Springer Berlin Heidelberg, 1993. — Vol. 700 of Lecture Notes in Computer Science. — P. 40–51.

[26] Hochbaum Dorit S. Approximating clique and biclique problems // Journal of Algorithms. — 1998. — Vol. 29, no. 1. — P. 174–200.

[27] Bliznets Ivan, Fomin Fedor, Pilipczuk Michał, Villanger Yngve. Largest Chordal and Interval Subgraphs Faster Than 2^n // Algorithms–ESA 2013. — Springer, 2013. — P. 193–204.

[28] Cai Leizhen. Fixed-parameter tractability of graph modification problems for hereditary properties // Information Processing Letters. — 1996. — Vol. 58, no. 4. — P. 171–176.

[29] Cao Yixin, Marx Dániel. Interval deletion is fixed-parameter tractable // ACM Transactions on Algorithms (TALG). — 2015. — Vol. 11, no. 3. — P. 21.

[30] Cao Yixin. Unit Interval Editing is Fixed-Parameter Tractable // Automata, Languages, and Programming / Ed. by Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, Bettina Speckmann. — Springer Berlin Heidelberg, 2015. — Vol. 9134 of Lecture Notes in Computer Science. — P. 306–317.

[31] Heggernes Pinar, van't Hof Pim, Jansen Bart. et al. Parameterized Complexity of Vertex Deletion into Perfect Graph Classes // Fundamentals of Computation Theory / Ed. by Olaf Owe, Martin Steffen, JanArne Telle. — Springer Berlin Heidelberg, 2011. — Vol. 6914 of Lecture Notes in Computer Science. — P. 240–251.

[32] Garey Michael R, Johnson David S. Computers and Intractability: A Guide to the Theory of NP-completeness. — 1979.

[33] Yannakakis Mihalis. Computing the minimum fill-in is NP-complete // SIAM Journal on Algebraic Discrete Methods. — 1981. — Vol. 2, no. 1. — P. 77–79.

[34] Fomin Fedor V, Kratsch Dieter, Todinca Ioan. Exact (exponential) algorithms for treewidth and minimum fill-in // Automata, Languages and Programming. — Springer, 2004. — P. 568–580.

[35] Fomin Fedor V, Kratsch Dieter, Todinca Ioan, Villanger Yngve. Exact algorithms for treewidth and minimum fill-in // SIAM Journal on Computing. — 2008. — Vol. 38, no. 3. — P. 1058–1079.

[36] Natanzon Assaf, Shamir Ron, Sharan Roded. A polynomial approximation algorithm for the minimum fill-in problem // SIAM Journal on Computing. — 2000. — Vol. 30, no. 4. — P. 1067–1079.

[37] Kaplan Haim, Shamir Ron, Tarjan Robert E. Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping // Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on / IEEE. — 1994. — P. 780–791.

[38] Fomin Fedor V, Villanger Yngve. Subexponential parameterized algorithm for minimum fill-in // SIAM Journal on Computing. — 2013. — Vol. 42, no. 6. — P. 2197–2216.

[39] Drange Pål Grønås, Fomin Fedor V, Pilipczuk Michał, Villanger Yngve. Exploring subexponential parameterized complexity of completion problems // arXiv preprint arXiv:1309.4022. — 2013.

[40] Bliznets Ivan, Fomin Fedor V, Pilipczuk Marcin, Pilipczuk Michał. A subexponential parameterized algorithm for Interval Completion // arXiv preprint arXiv:1402.3473. — 2014.

[41] Bliznets Ivan, Fomin Fedor V, Pilipczuk Marcin, Pilipczuk Michał. A subexponential parameterized algorithm for Proper Interval Completion // Algorithms-ESA 2014. — Springer, 2014. — P. 173–184.

[42] Impagliazzo Russell, Paturi Ramamohan. Complexity of k-SAT // Proceedings of the 14th Annual IEEE Conference on Computational Complexity, Atlanta, Georgia, USA, May 4-6, 1999. — 1999. — P. 237–240. — URL: http://doi.ieeecomputersociety.org/10.1109/CCC.1999.766282.

[43] Cao Yixin, Marx Dániel. Chordal Editing is Fixed-Parameter Tractable // Algorithmica. — 2015. — P. 1–20.

[44] Bliznets Ivan, Fomin Fedor, Pilipczuk Michał, Villanger Yngve. Largest Chordal and Interval Subgraphs Faster than 2^n // Algorithmica. — 2015. — P. 1–26.

[45] Bliznets Ivan, Cygan Marek, Komosa Pawel et al. Lower bounds for the parameterized complexity of Minimum Fill-In and other completion problems // Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms. — SIAM, 2016. — P. 1132–1151. — http://epubs.siam.org/doi/pdf/10.1137/1.9781611974331.ch79.

[46] Impagliazzo Russell, Paturi Ramamohan, Zane Francis. Which Problems Have Strongly Exponential Complexity? // J. Comput. Syst. Sci. — 2001. — Vol. 63, no. 4. — P. 512–530.

[47] Impagliazzo Russell, Paturi Ramamohan. On the Complexity of k-SAT // J. Comput. Syst. Sci. — 2001. — Vol. 62, no. 2. — P. 367–375.

[48] Lokshtanov Daniel, Marx Dániel, Saurabh Saket. Lower bounds based on the Exponential Time Hypothesis // Bulletin of the EATCS. — 2011. — Vol. 105. — P. 41–72.

[49] Gilbert John R., Rose Donald J., Edenbrandt Anders. A Separator Theorem for Chordal Graphs // SIAM Journal on Algebraic Discrete Methods. — 1984. — Vol. 5, no. 3. — P. 306–313.

[50] Cygan Marek, Fomin Fedor V., Kowalik Łukasz et al. Parameterized Algorithms. — Springer, 2015. — In print.

[51] Fomin Fedor V, Villanger Yngve. Treewidth computation and extremal combinatorics // Combinatorica. — 2012. — Vol. 32, no. 3. — P. 289–308.

[52] Schroeppel Richard, Shamir Adi. A $t = O(2^{n/2}), s = O(2^{n/4})$ Algorithm for certain NP-Complete problems // SIAM Journal on Computing. — 1981. — Vol. 10, no. 3. — P. 456–464.

[53] Golumbic Martin C. Algorithmic Graph Theory and Perfect Graphs. 1980.

[54] Arora Sanjeev, Barak Boaz. Computational Complexity - A Modern Approach. — Cambridge University Press, 2009. — ISBN: 978-0-521-42426-4. — URL: http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.

[55] Schaefer Thomas J. The complexity of satisfiability problems // Proceedings of the tenth annual ACM symposium on Theory of computing / ACM. — 1978. — P. 216–226.

[56] Garey M. R., Johnson David S., Stockmeyer Larry J. Some Simplified NP-Complete Graph Problems // Theor. Comput. Sci. — 1976. — Vol. 1, no. 3. — P. 237–267.

[57] Yannakakis Mihalis. Computing the Minimum Fill-In is NP-Complete // SIAM Journal on Algebraic Discrete Methods. — 1981. — Vol. 2, no. 1. — P. 77–79.